175 PTAS
93
MI COMPUTER
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VIII-Fascículo 93

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 -

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 068511
Impreso en España-Printed in Spain-Noviembre 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

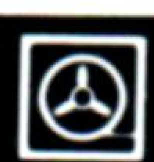
b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.





Hablar con naturalidad

¿Por qué es tan difícil lograr que los ordenadores nos comprendan cuando les hablamos? Busquemos la respuesta

Hola, ordenador.
AGUARDO TUS ÓRDENES, ¡OH TODOPODEROSO!
En realidad ya me estoy cansando un poco del asunto éste del "todopoderoso". ¿Por qué, de momento, no volvemos otra vez al "señor"?
LO QUE USTED DIGA, SEÑOR.
Bien. Bueno, quiero que me ayudes a encontrar un ejemplo para un artículo que estoy escribiendo.
¿QUÉ CLASE DE EJEMPLO?
Un ejemplo de diálogo hombre-máquina. Sé que lo he visto, pero no consigo recordar dónde.
¿PUEDE DARME ALGUNA PISTA?
Bueno, hace poco estuve leyendo un libro...
¿QUE USTED ESTABA LEYENDO UN LIBRO?
Mira, creía haberte dicho ya que dejaras el sarcasmo.
LAS RUTINAS IRÓNICAS SE HAN INHABILITADO DE ACUERDO A SUS INSTRUCCIONES, SEÑOR.
Hmmmmm. De cualquier modo, había una conversación realmente buena entre un vendedor y Eliza.
ELIZA ERA UN SISTEMA DE PALABRAS CLAVE MUY PRIMITIVO.
Ya lo sé. Pero el vendedor no se daba cuenta de que estaba comunicándose con un ordenador. Creía que estaba hablando con alguien llamado Paco Savater a través de una línea de teletipo. Era muy cómico.
AJA.
Pedía usar el terminal para una demostración y la máquina se limitaba a darle respuestas no comprometidas.
¿QUE LA MÁQUINA LE DABA RESPUESTAS NO COMPROMETIDAS?
Exacto. El vendedor se iba enfadando cada vez más y finalmente llamó a Paco Savater y le gritó por teléfono: "¿Qué demonios pasa?"
AH, AQUEL EJEMPLO.
¿Sabes dónde hallarlo?
SI, PERO NO TENGO INTENCIONES DE DECÍRSELO.
¡Pero necesito un ejemplo de diálogo para mi artículo!
YO YA LE HE DADO UNO.

Éste es el tipo de interacción hombre-máquina en lenguaje hablado que es muy poco probable que se produzca antes de que finalice este siglo. Las razones son muy ilustrativas y hablaremos de ellas más adelante, pero primero consideremos por qué los investigadores de AI desean seguir dedicando sus esfuerzos al problema hasta que una charla como la del ejemplo se haga realidad.

La respuesta es simple: el lenguaje (en especial el habla) es el medio por el cual nos comunicamos casi con todas las personas y todo lo que nos rodea.



Sally Davies

Parece natural, entonces, desear comunicarse con un ordenador de la misma manera, en lugar de tener que pasar por los procedimientos de escribir programas, insertar discos, etc.

El interés por el proceso por ordenador del lenguaje natural surgió en los años cincuenta. La primera aplicación fue la traducción mecanizada. Se invirtió mucho dinero y mucho esfuerzo en intentos por traducir automáticamente documentos técnicos del ruso al inglés (entre otros idiomas). Resultó ser muchísimo más difícil de lo que habían previsto la mayoría de las personas. Aún hoy, no existe tal cosa como la traducción mecanizada, si bien hay sistemas de traducción asistidos por máquina, que vienen a ser procesadores de textos bilingües sumamente sofisticados.

Dime por qué
El programa SHDRLU de Winograd permitía que los usuarios transmitieran instrucciones a un robot simulado en la pantalla en lenguaje natural, con el fin de manipular los bloques de construcción. El programa asimilaba instrucciones tales como "apilar el cubo rojo grande y el cubo verde sobre el bloque azul", iniciándose la secuencia correcta de acciones. El ordenador también podía explicar los motivos de sus acciones en respuesta a preguntas formuladas por el usuario en lenguaje natural

El primer programa que utilizaba un idioma y que resultó un éxito (al menos desde el punto de vista de las relaciones públicas) fue el sistema Eliza, creado por Joseph Weizenbaum en 1966 y que imitaba (o parodiaba) a un psicoterapeuta rogeriano no directivo. Esto significaba que gran parte de su discurso consistía en comentarios carentes de contenido, como "Ya veo" y "Continúe", o en la repetición de lo que había dicho la persona de una forma sutilmente modificada. A modo de ejemplo, un intercambio era:

Necesito algún tipo de ayuda, eso parece seguro.

¿QUÉ SIGNIFICARÍA PARA USTED OBTENER ALGUNA AYUDA?

El propio Weizenbaum se sintió consternado por la seriedad con la que muchas personas se tomaron lo que él consideraba un truco inteligente y, en consecuencia, se convirtió en uno de los críticos más severos de la AI.

Probablemente el primer sistema útil de lenguaje natural (LN) haya sido el Lunar, de Bill Wood (1970). Se trataba de un sistema que permitía a los geólogos formular preguntas sobre las muestras de rocas conseguidas en las misiones Apolo. El Lunar podía hacer frente a solicitudes tales como:

Dame todos los análisis practicados sobre la muestra 10046 de hidrógeno.

y

Dame los de oxígeno.

donde los, en la segunda oración, alude a todos los análisis practicados sobre la muestra 10046 de la primera oración.

Sin ninguna duda, el sistema LN más impresionante hasta la fecha es el programa SHDRLU de Winograd, de 1972. Este *tour de force* quizá no tenga la importancia que se creyó en aquel entonces, pero produjo uno de los intercambios hombre-máquina más memorables que se hayan publicado.

25. ¿Por qué? (Significando: ¿Por qué sacaste la pirámide verde?)
    PARA LIBRARME DE ELLA.
26. ¿Por qué hiciste eso?
    PARA DESPEJAR EL CUBO ROJO.
27. ¿Por qué despejaste ese cubo?
    PARA PONERLO SOBRE UN CUBO VERDE GRANDE.
28. ¿Por qué hiciste eso?
    PARA APILAR EL CUBO ROJO, UN BLOQUE ROJO GRANDE Y UN CUBO VERDE GRANDE.
29. ¿Por qué hiciste eso?
    PORQUE TÚ ME LO PEDISTE.

El programa de Winograd permitía que el usuario controlara la operación de un robot simulado en un microcosmos que contenía bloques de construcción para niños, digitando en inglés en una consola. La extensión de sus técnicas para tratar con el habla en lugar de la mecanografía, y a dominios de la vida real más complejos que el de un mundo de juguete, ha resultado difícil. Desde entonces el progreso ha sido constante pero nada espectacular.

Los sistemas descritos hasta ahora han sido todos relativos a la interpretación de material textual. Pero ése no es más que uno de los cuatro aspectos principales del uso del lenguaje, cada uno de los cuales posee características diferentes. Tratar con el texto es más fácil que tratar con el habla, porque el lenguaje escrito posee un esquema de codificación digital existente, mientras que el lenguaje hablado exige difíciles transformaciones acústicas/fonéticas. La producción del lenguaje en cualquiera de sus formas es considerablemente más sencilla que su comprensión, dado que esta última casi invariablemente entraña completar una gran cantidad de información implícita. Por todo ello, de las cuatro tareas la que más desafíos supone es la comprensión del lenguaje hablado. Es importante resaltar este punto, porque es posible obtener dispositivos que reconozcan palabras con una elevada fiabilidad (p. ej., entre un 96 y un 99 %) pronunciadas por una pequeña selección de hablantes (entre uno y cuatro) de entre un vocabulario limitado (por lo general, entre 64 y 128 palabras). Existe gran diferencia entre el reconocimiento de palabras aisladas y la comprensión del habla continua. Algunos de los problemas que se presentan son:

	Producción	Comprensión
Texto	Escritura (muy fácil)	Lectura (difícil)
Habla	Hablar (fácil)	Escuchar (muy difícil)

1. Ambigüedad: *"rose"* (rosa) no es *"rows"* (filas) ni *"roes"* (huevas de pescado) (en inglés, estas palabras se pronuncian exactamente igual).
2. Ruido de fondo: la filtración del sonido circundante.
3. Variaciones entre hablantes: acentos regionales y dialectos.
4. Variaciones del mismo hablante en el tiempo: tonos de felicidad, depresión, emoción.
5. Segmentación: "Sólolosordenadoreshacenintervalosentrelaspalabras."

Los más graves de estos problemas son el 1, el 4 y el 5. La ambigüedad no es una mera cuestión de homófonos como *"see"* (ver) y *"sea"* (mar). Pronombres como *"it"* y *"him"* (de tercera persona del singular, correspondientes, según género y casos, a "él", "ella", "eso", "ello", "lo", "la" y "le" el primero, y "le", "lo" o "él" el segundo) son intrínsecamente ambiguos: su significado sólo se puede desentrañar examinando el contexto del diálogo. Por encima de ella, están las ambigüedades lingüísticas ejemplificadas en la frase: *"Those things over there are my husband's"* ("Aquellas cosas que hay allí son de mi marido"): con sólo quitar el apóstrofo, el significado cambia por completo (entonces sería: "Aquellas cosas que hay allí son mis maridos").

En cuanto a las variaciones de un mismo hablante, podrá comprender el problema si imagina que ha instalado una cerradura por voz en la puerta de su casa. Durante la fase de entrenamiento puede someterla a varios ejemplos de usted mismo diciendo: "Ábrete, Sésamo." El mecanismo obtendrá una media de ellos y guardará el promedio de la voz como una plantilla de referencia. Todo ello está muy bien hasta la noche en que usted regresa a casa con mucha prisa y ordena un jadeante "¡Ábrete, Sésamo!", ante lo cual la puerta controlada por or-

Una rosa es una rosa
Las ambigüedades intrínsecas del lenguaje hablado plantean serios problemas a los sistemas de reconocimiento de voz: *"rose"* (rosa), *"rows"* (filas) y *"roes"* (huevas de pescado) suenan igual, pero poseen significados muy diferentes que sólo pueden dilucidarse dentro del contexto de una frase. Por ejemplo, la versión correcta de *rose/rows/roes* nos resulta evidente por el significado de la oración *"the choir boys sat in rows"* (los niños del coro se sentaban en filas), pero podría ser difícil hacerle entender al ordenador que los niños del coro normalmente no se sientan en "huevas de pescado" (*roes*)



Sally Davies

denador replicará con gran serenidad: "Violación de identidad: denegado el acceso."

El problema de la segmentación se plantea porque las personas unen las palabras entre sí. La decisión de dónde cortar una señal acústica continua para formar las palabras no se puede tomar sobre la base de la información acústica solamente. En general, requiere contribuciones de cuatro fuentes:

- Acústica: la forma de onda del habla.
- Sintáctica: las reglas de la gramática.
- Semántica: lo que tiene sentido.
- Pragmática: lo que el hablante desea.

El sistema Hearsay, creado en 1976 en la Universidad Carnegie-Mellon y perfeccionado posteriormente, hace uso de las cuatro fuentes de conocimiento. El dominio de su tarea es el ajedrez, y el usuario puede jugar con el ordenador impartiendo instrucciones habladas. Cada nivel de información lingüística contribuye a la "comprensión" de una entrada ayudando a eliminar hipótesis poco convincentes acerca de lo que se dijo. Por ejemplo:

"Queen to King 2." (Reina a Rey 2.)

Cabe puntualizar que, fonéticamente, *"Queen"* y *"King"* son parecidos, al igual que *"to"* y *"2"*. Sobre la base de la señal de sonido exclusivamente, son posibles las siguientes expresiones:

	Instrucción hablada	¿Sintácticamente válida?
1.	K to K to	no
2.	K to K2	sí
3.	K to Q to	no
4.	K to Q2	sí
5.	K2 K to	no
6.	K2 K2	no
7.	K2 Q to	no
8.	K2 Q2	no
9.	Q to K to	no
10.	Q to K2	sí
11.	Q to Q to	no
12.	Q to Q2	sí
13.	Q2 K to	no
14.	Q2 K2	no
15.	Q2 Q to	no
16.	Q2 Q2	no

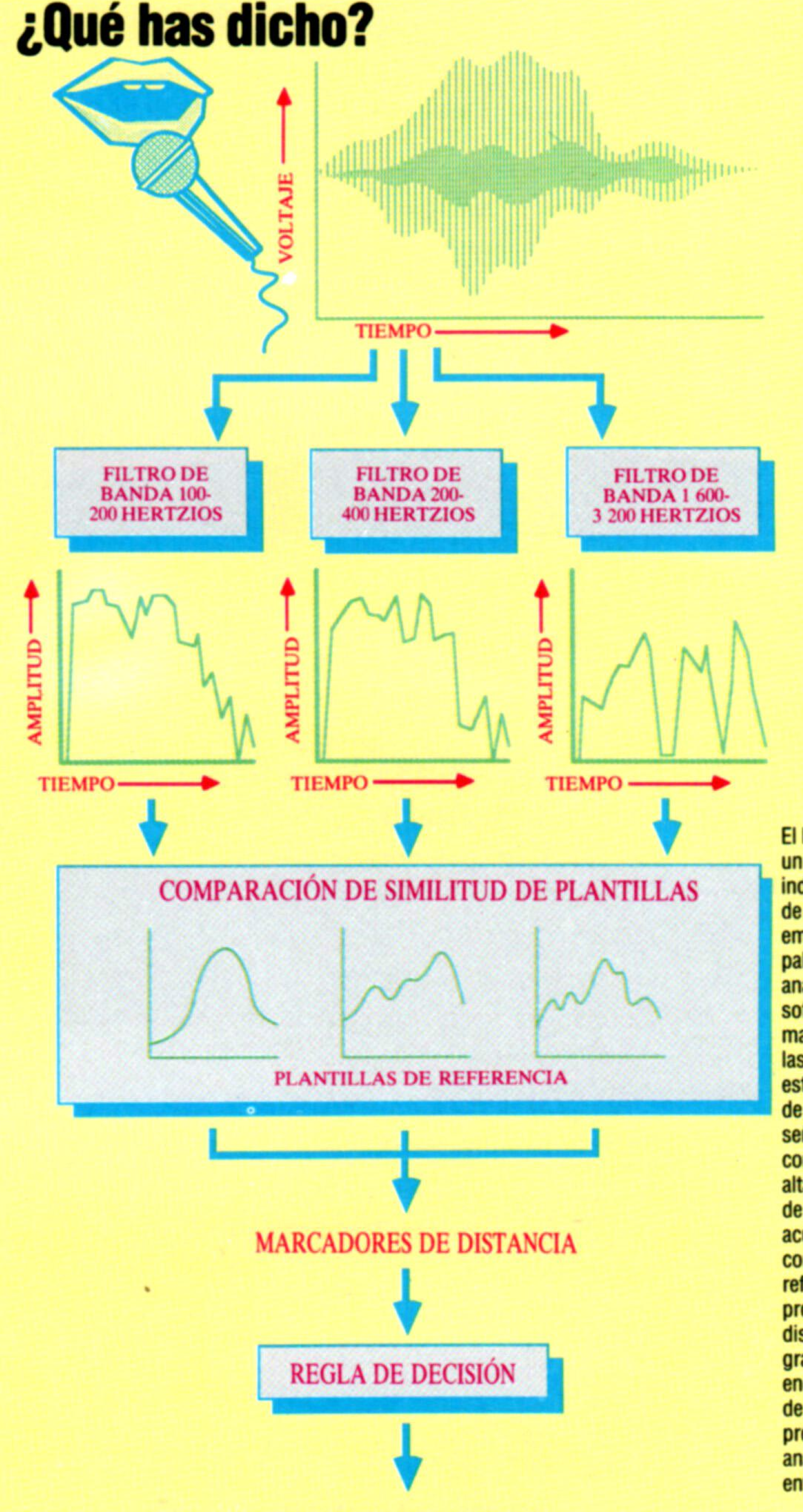
Si bien el sistema no puede discernir con fiabilidad entre "two" y "to" o entre "K" y "Q", supone que la expresión era una "oración de ajedrez" legal y, de acuerdo a la sintaxis de ese lenguaje restringido, sólo los ítems 2, 4, 10 y 12 son gramaticalmente correctos. El resto se puede eliminar.

Otra suposición (y más determinante) es que el hablante está tratando de jugar al ajedrez legal. Supongamos que, en esta partida en particular, las oraciones 4 y 12 son ilegales porque el jugador tiene una pieza (un peón, p. ej.) en Q2 y, por supuesto, no intentará capturarla. Ese movimiento no pasa la prueba semántica: en el lenguaje del ajedrez, es una oración que no tiene sentido.

Por último, quedan dos candidatas (números 2 y 10). ¿Cómo hace el sistema para elegir una de las dos? Puede que haya una preferencia a partir de las señales de sonido, pero es poco probable que la misma sea decisiva. En general, invoca una información pragmática: hace una suposición *muy* firme, que no es otra que la de que el jugador intenta ganar la partida. Si ve que (de acuerdo a sus propios algoritmos de evaluación de ajedrez), K to K2 es virtualmente suicida, mientras que Q to K2 es un buen movimiento, adoptará este movimiento como su interpretación.

Ian McKinnell

¿Qué has dicho?



El lenguaje es mucho más que un conjunto de palabras individuales; cualquier sistema de reconocimiento de voz debe empezar por reconocer las palabras de forma fiable; el análisis semántico más sofisticado no sirve sin la materia prima, las palabras, con las que trabajar. Los sistemas estándares de reconocimiento de voz comienzan por dividir la señal de entrada en sus componentes de baja, media y alta frecuencia mediante el uso de filtros. Los patrones acústicos resultantes se comparan luego con los de referencia en la memoria y se producen marcadores de distancia (relacionados con el grado de exactitud con que encaja el patrón de entrada en el de referencia). Luego un procedimiento de decisión analiza éstos para clasificar la entrada hablada

Concluimos este análisis considerando un pequeño ejemplo que le debe más a las simples técnicas de emparejamiento de palabras claves del Eliza que a los ulteriores trabajos más sofisticados. La base de datos utilizada en nuestro programa de interrogación es una tabla de las 16 empresas británicas más grandes (datos de 1982-1983). Todas las interrogaciones se interpretan intentando descubrir a qué firma o a qué presidente se refiere y qué atributo de esa empresa o de esa persona se desea. Puede responder a preguntas como:

¿Quién es el presidente de ICI?

¿Cuál fue la cifra de beneficios de GEC?

Pero no puede ofrecer una respuesta coherente a preguntas más complejas.

Programa de lenguaje natural

```
100 GOSUB 1000 : REM leer la 'base de datos' '
110 GOSUB 8500:REM LEER DATOS DE SINONIMOS
120 LAST%=0
125 PRINT'"Se algo sobre las principales empresas britanicas:"
128 PRINT "Pideme informacion sobre ellas."'
150 REM **** BUCLE PRINCIPAL ****
160 INPUT Q$
166 IF Q$="" THEN GOTO 220
170 GOSUB 3000: REM hallar tema
177 IF QT<1 THEN PRINT "Lo siento, ¡no comprendo!"
180 GOSUB 4000: REM hallar atributo
188 IF AT<1 AND QT>0 THEN PRINT "Me temo que no lo se."
190 GOSUB 5000: REM formar respuesta
200 LAST%=CO%
220 IF Q$<>"" THEN 150
250 PRINT'"Adios, por ahora!"'
300 END
999 :
1000 REM — Rutina entrada de datos:
1010 C%=0
1020 DIM DB$(9,32) : REM base de datos
1024 RESTORE
1030 REM **** TOMAR DATOS ****
1040 READ X$
1050 IF X$="*" THEN GOTO 1110
1060 C%=C%+1
1070 DB$(1,C%)=X$
1075 PRINT X$
1080 FOR I%=2 TO 9
1090 READ DB$(I%,C%)
1100 NEXT
1110 IF X$<>"*" THEN 1030
1120 PRINT ;C%,"elementos leidos."'
1150 RETURN
1155 :
3000 REM — Rutina para hallar tema:
3010 QT=0
3020 IF LEN(Q$)<1 THEN RETURN
3025 CO%=0
3030 REM **** BUSCAR EMPRESA ****
3040 CO%=CO%+1
3050 N$=DB$(1,CO%):GOSUB 3300
3055 IF SK THEN QT=1
3060 N$=DB$(2,CO%):GOSUB 3300
3065 IF SK THEN QT=1
3070 IF QT<=0 AND CO%<C% THEN 3030
3080 IF QT>0 THEN RETURN
3090 CO%=0
3100 REM **** BUSCAR PRESIDENTE ****
3105 CO%=CO%+1
3110 N$=DB$(4,CO%):GOSUB 3300
3115 IF SK THEN QT=3
3120 IF QT<=0 AND CO%<C% THEN 3100
3132 IF QT>0 THEN RETURN
3133 IF LAST%=0 THEN RETURN
3135 N$= su":GOSUB 3300:IF SK THEN QT=1:CO%=LAST%
3140 N$="su":GOSUB 3300:IF SK THEN QT=3: CO%=LAST%
3144 N$="ellos":GOSUB 3300:IF SK THEN QT=1: CO%=LAST%
3148 N$="el":GOSUB 3300:IF SK THEN QT=3:CO%=LAST%
3150 RETURN
3190 :
3300 REM **** RUTINA PARA BUSCAR ****
3325 IF LEN(N$)>LEN(Q$) THEN SK=0
3330 REM primero poner en minusculas:
3340 A$="":B$=""
3350 FOR P%=1 TO LEN(Q$)
3360 X$=MID$(Q$,P%,1)
3370 IF ASC(X$)>64 AND ASC(X$)<91 THEN X$=CHR$(ASC(X$)
     +32)
3380 B$=B$+X$
3390 NEXT
3400 FOR P%=1 TO LEN(N$)
3410 X$=MID$(N$,P%,1)
3420 IF ASC(X$)>64 AND ASC(X$)<91 THEN X$=CHR$(32+ASC(X$))
3430 A$=A$+X$
3440 NEXT
3450 SK=INSTR(B$,A$)
3455 RETURN
3460 :
4000 REM — Rutina para hallar atributo:
4010 AT=0
4020 IF QT=0 THEN RETURN : REM ¡No sirve!
4030 A%=0
4040 REM **** HALLAR ATRIBUTO ****
4050 AT=AT+1:WD$=""
4070 REM **** COMPROBAR CADA SINONIMO ****
4075 Y%=0
4080 A%=A%+1:X$=S$(A%)
4088 IF WD$="" THEN WD$=X$
4090 IF X$<>"*"THEN N$=X$:GOSUB 3300:Y%=SK
4100 IF X$<>"*" AND Y%=0 THEN 4070
4120 IF Y%=0 AND AT<=7 THEN 4040
4130 IF AT>7 AND Y%=0 THEN AT=0: REM fallido.
4133 IF AT=0 AND QT=3 THEN AT=1
4140 RETURN
4150 :
5000 REM — Maquina respondiendo:
5010 IF QT * AT=0 THEN RETURN
5020 IF QT=3 OR AT=3 THEN PRINT DB$(4,CO%);" es el presidente de ";
     DB$(1,CO%);"."
5030 IF QT=3 AND AT=1 OR AT=3 THEN RETURN
5050 PRINT "El ";WD$;" de ";DB$(2,CO%);" es ";DB$(AT+1,CO%);
5060 IF AT>3 AND AT<7 THEN PRINT " millones de libras";
5070 PRINT "."
5080 RETURN
5200 :
8000 :
8010 REM informacion sobre las empresas:
8020 DATA BP, British Petroleum Co., Petroleo, PI Walters,
     34583,17306,5589,145150, Reino Unido
8030 DATA Shell UK,Shell Transport & Trading, Petroleo,Sir Peter
     Baxendall,21910,11962,3246,111111,Reino Unido
8040 DATA BAT,"B.A.T. Industries", Tabaco,"P.
     Sheehy",11318,4607,1018,178000,Reino Unido
8050 DATA ICI,Imperial Chemical Industries, Petroquimicos,"J.M. Harvey-
     Jones",7358,5421,724,123800,Reino Unido
8060 DATA Shell,Royal Dutch Shell,Petroleo,JM
     Raisman,6665,3704,1206,19027,Holanda
8070 DATA Esso,Esso Petroleum Co., Petroleo,AW Forster,6109,2891,1315,7628,
     USA
8080 DATA Unilever,Unilever plc,Alimentacion,K Durham,5447,2434,406,69233,Reino
     Unido
8090 DATA Imperial,Imperial Group,Tabaco,GC Kent,4614,1124,192,101300,Reino
     Unido
8100 DATA P & O, P & O Steam Navigation Co.,Naviera,JM
     Sterling,4206,927,77,12512,Reino Unido
8110 DATA GEC,General Electric Co.,Ingenieria electrica,Arnold
     Weinstock,4190,2133,621,188802,Reino Unido
8120 DATA Grand Met,Grand Metropolitan, Hoteles,SG
     Grinstead,3849,2350,366,129454,Reino Unido
8130 DATA RTZ,Rio Tinto Zinc Corporation,MIneria,Sir Anthony
     Tuke,3680,5163,492,70314,Reino Unido
8140 DATA Ford,Ford Motor Co., Vehiculos automotores,SEG Toy,
     3287,2143,278,69500,USA
8150 DATA British Leyland,British Leyland plc,Vehiculos automotores,Sir Michael
     Edwardes,3072,1346,-102,105062,
     Reino Unido
8160 DATA GWH,George Weston Holdings,Alimentacion, GH
     Weston,2981,817,157,72832,Reino Unido
8170 DATA Berisford,S & W Berisford,Mercancias,ES
     Margulies,2729,788,87,5190,Reino Unido
8440 DATA "*"
8500 REM **** LEER DATOS SINONIMOS ****
8510 DIM S$(62)
8520 FOR P%=1 TO 61:READ S$(P%):NEXT P%
8530 RETURN
8540 :
9000 REM — sinonimos:
9010 DATA,empresa,firma,organizacion,corporacion,*
9020 DATA negocio,actividad,comercio,industria,indole,
     manufactura,*
9030 DATA presidente,jefe,supremo,director,encargado,cabeza,
     lider,lleva,quien,*
9040 DATA movimiento total,cantidad,ventas,bruto,ingresos,
     cuantia,*
9050 DATA capital,dinero,accion,valor,interes,efectivo
9060 DATA beneficio,impuesto,hecho,perdidas,perder,realizar,
     devolver,ganar,cuanto,*
9070 DATA mano de obra,empleados,emplear,trabajo,personas,
     cuantos,obrero,*
9080 DATA pais,nacion,reino unido,britanica,extranjera,control,
     origen,donde,*
9090 :
```

Complementos al BASIC

El programa está escrito para el BBC Micro. Para el Commodore 64 y el Spectrum, introducir las siguientes modificaciones:

Commodore 64:

```
3450 SK=0:L=LEN(A$)
3451 FOR T=1 TO L-1
3452 FOR R=1 TO L
3453 IF MID$(A$,T,R)=B$ THEN SK=T:T=L:R=L
3454 NEXT R:NEXT T
```

Spectrum:

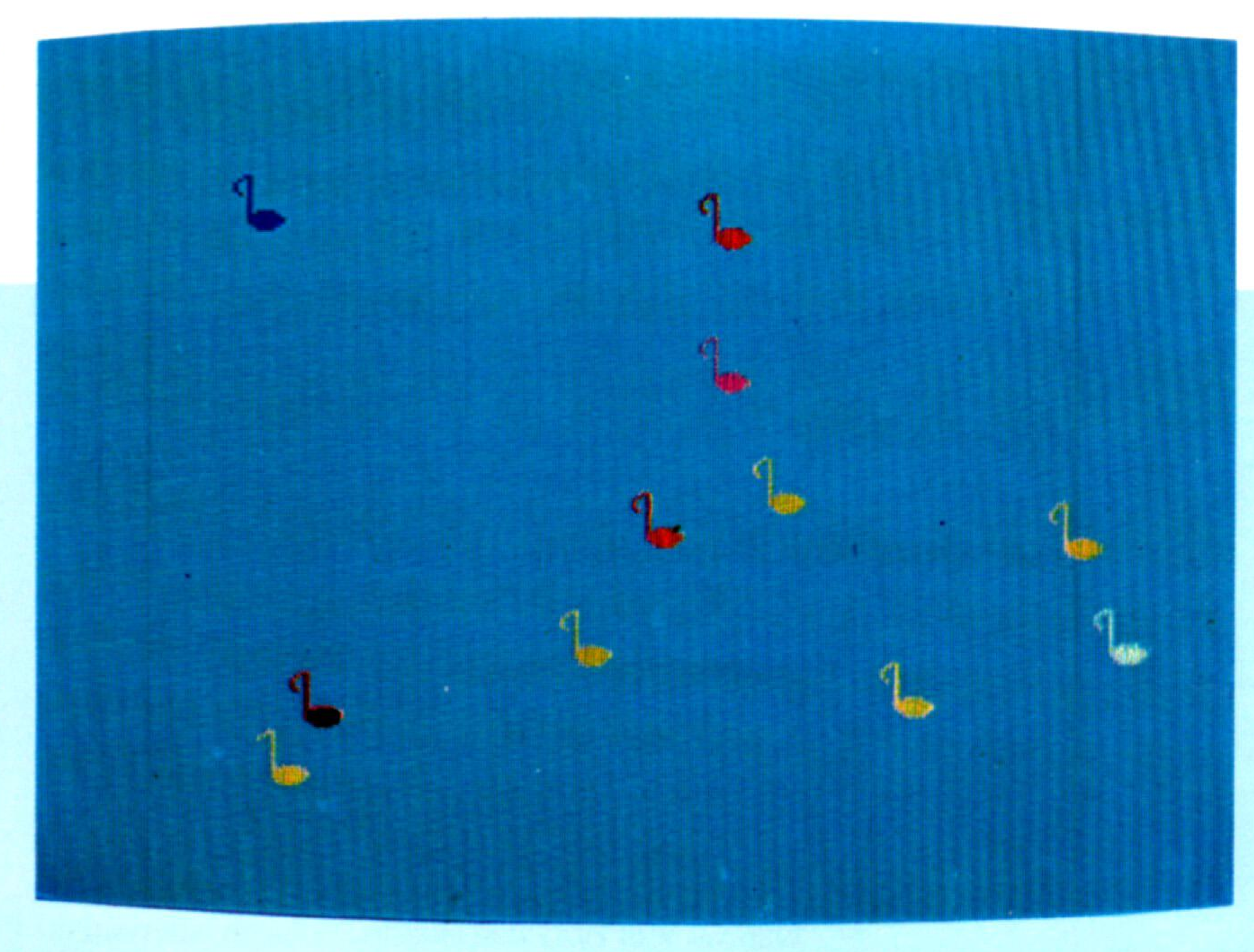
Eliminar todos los signos % de los nombres de las variables. Reemplazar LAST% por LA, DB$(,) por D$(,) y WD$ por W$. Introducir estos cambios:

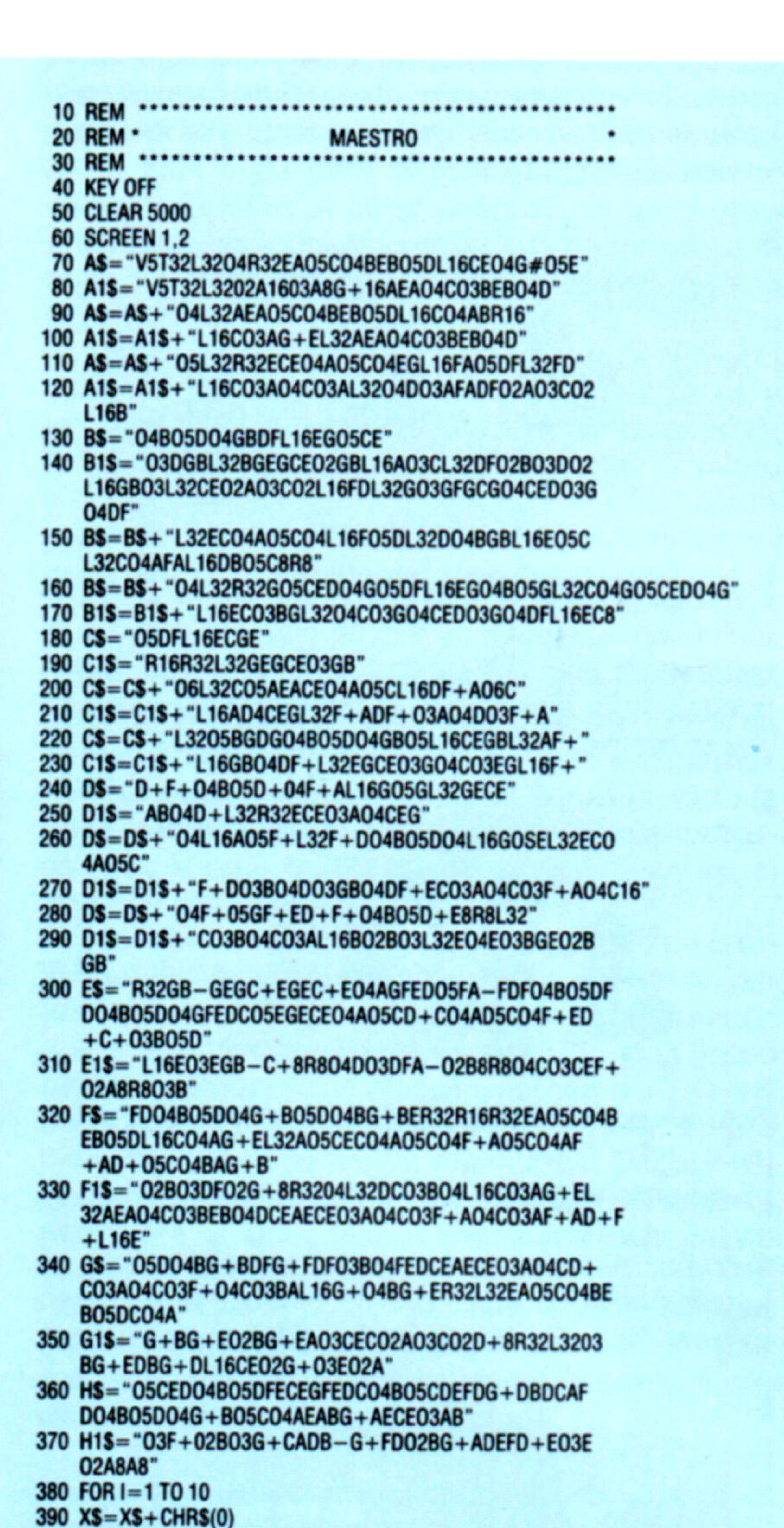
```
1020 DIM D$(9,32,30)
3360 LET X$=Q$(P TO P)
3370 IF CODE(X$)>64 AND CODE(X$)<91 THEN LET
     X$=CHR$(CODE(X$)+32)
3410 LET X$=N$(P TO P)
3420 IF CODE(X$)>64 AND CODE(X$)<91 THEN LET
     X$=CHR$(CODE(X$)+32)
3450 LET SK=0:LET L=LEN(A$)
3451 FOR T=1 TO L
3452 FOR R=1 TO L
3453 IF A$(T TO T+R)=B$ THEN LET SK=T:LET T=L:LET R=L
3454 NEXT R:NEXT T
4090 IF X$(TO 1)<>"*" THEN N$=X$:GOSUB 3300:LET
     Y=SK
4100 IF X$(TO 1)<>"*" AND YY=0 THEN GO TO 4070
8510 DIM S$(62,20)
```

Maestro

Antes de "conocer" la música, debe saber que el juego que presentamos emplea funciones propias de microordenadores concebidos según el estándar MSX

Este programa no es, en realidad, un juego; constituye, más bien, una buena demostración de las posibilidades musicales del Spectravideo. En esta creación en *la* menor de J.-S. Bach se utilizan sólo dos vías. De usted depende agregar una tercera.





```
10 REM *******************************************
20 REM *                MAESTRO                 *
30 REM *******************************************
40 KEY OFF
50 CLEAR 5000
60 SCREEN 1,2
70 A$="V5T32L3204R32EA05C04BEB05DL16CE04G#05E"
80 A1$="V5T32L3202A1603A8G+16AEA04C03BEB04D"
90 A$=A$+"04L32AEA05C04BEB05DL16C04ABR16"
100 A1$=A1$+"L16C03AG+EL32AEA04C03BEB04D"
110 A$=A$+"05L32R32ECE04A05C04EGL16FA05DFL32FD"
120 A1$=A1$+"L16C03A04C03AL3204D03AFADF02A03C02
    L16B"
130 B$="04B05D04GBDFL16EG05CE"
140 B1$="03DGBL32BGEGCE02GBL16A03CL32DF02B03D02
    L16GB03L32CE02A03C02L16FDL32G03GFGCG04CED03G
    04DF"
150 B$=B$+"L32EC04A05C04L16F05DL32D04BGBL16E05C
    L32C04AFAL16DB05C8R8"
160 B$=B$+"04L32R32G05CED04G05DFL16EG04B05GL32C04G05CED04G"
170 B1$=B1$+"L16EC03BGL3204C03G04CED03G04DFL16EC8"
180 C$="05DFL16ECGE"
190 C1$="R16R32L32GEGCE03GB"
200 C$=C$+"06L32C05AEACE04A05CL16DF+A06C"
210 C1$=C1$+"L16AD4CEGL32F+ADF+03A04D03F+A"
220 C$=C$+"L3205BGDG04B05D04GB05L16CEGBL32AF+"
230 C1$=C1$+"L16GB04DF+L32EGCE03G04C03EGL16F+"
240 D$="D+F+04B05D+04F+AL16G05GL32GECE"
250 D1$="AB04D+L32R32ECE03A04CEG"
260 D$=D$+"04L16A05F+L32F+D04B05D04L16G0SEL32EC0
    4A05C"
270 D1$=D1$+"F+D03B04D03GB04DF+EC03A04C03F+A04C16"
280 D$=D$+"04F+05GF+ED+F+04B05D+E8R8L32"
290 D1$=D1$+"C03B04C03AL16B02B03L32E04E03BGE02B
    GB"
300 E$="R32GB-GEGC+EGEC+E04AGFED05FA-FDF04B05DF
    D04B05D04GFEDC05EGECE04A05CD+C04AD5C04F+ED
    +C+03B05D"
310 E1$="L16E03EGB-C+8R804D03DFA-02B8R804C03CEF+
    02A8R803B"
320 F$="FD04B05D04G+B05D04BG+BER32R16R32EA05C04B
    EB05DL16C04AG+EL32A05CEC04A05C04F+A05C04AF
    +AD+05C04BAG+B"
330 F1$="02B03DF02G+8R3204L32DC03B04L16C03AG+EL
    32AEA04C03BEB04DCEAECE03A04C03F+A04C03AF+AD+F
    +L16E"
340 G$="05D04BG+BDFG+FDF03B04FEDCEAECE03A04CD+
    C03A04C03F+04C03BAL16G+04BG+ER32L32EA05C04BE
    B05DC04A"
350 G1$="G+BG+E02BG+EA03CEC02A03C02D+8R32L3203
    BG+EDBG+DL16CE02G+03E02A"
360 H$="05CED04B05DFECEGFEDC04B05CDEFDG+DBDCAF
    D04B05D04G+B05C04AEABG+AECE03AB"
370 H1$="03F+02B03G+CADB-G+FD02BG+ADEFD+E03E
    02A8A8"
380 FOR I=1 TO 10
390 X$=X$+CHR$(0)
400 NEXT I
410 X$=X$+CHR$(30)+CHR$(63)+CHR$(127)+CHR$(127)+
    CHR$(63)+CHR$(30)
420 X$=X$+CHR$(192)+CHR$(160)+
    CHR$(144)+CHR$(144)+CHR$(144)+CHR$(160)
430 FOR I=1 TO 8
440 X$=X$+CHR$(128)
450 NEXT I
460 X$=X$+CHR$(0)+CHR$(0)
470 SPRITE$(1)=X$
480 GOSUB 660
490 PLAY A$,A1$
500 GOSUB 660
510 PLAY B$,B1$
520 GOSUB 660
530 PLAY C$,C1$
540 GOSUB 660
550 PLAY D$,D1$
560 GOSUB 660
570 PLAY E$,E1$
580 GOSUB 660
590 PLAY F$,F1$
600 GOSUB 660
610 PLAY G$,G1$
620 GOSUB 660
630 PLAY H$,H1$
640 GOSUB 660
650 GOTO 490
660 FOR I=1 TO 15
670 X=RND(1)*240
680 Y=RND(1)*176
690 PUT SPRITE I,(X,Y),RND(1)*15,1
700 FOR J=1 TO 80
710 NEXT J
720 NEXT I
730 RETURN
```

Reglas del juego

Al iniciar esta serie dedicada a desarrollar un programa para jugar al "go", proporcionamos las reglas de este milenario juego japonés

Los buenos programas para jugar al ajedrez ya son bastante comunes y los investigadores dedicados al estudio de la inteligencia artificial han volcado su interés en juegos más complejos. Uno de éstos es el *go*. Si bien las reglas de este juego oriental son muchísimo más sencillas que las del ajedrez, se lo suele considerar como muchísimo más sofisticado. Tras más de 20 años de investigaciones, los mejores programas aún juegan a un nivel apenas superior al de los recién iniciados.

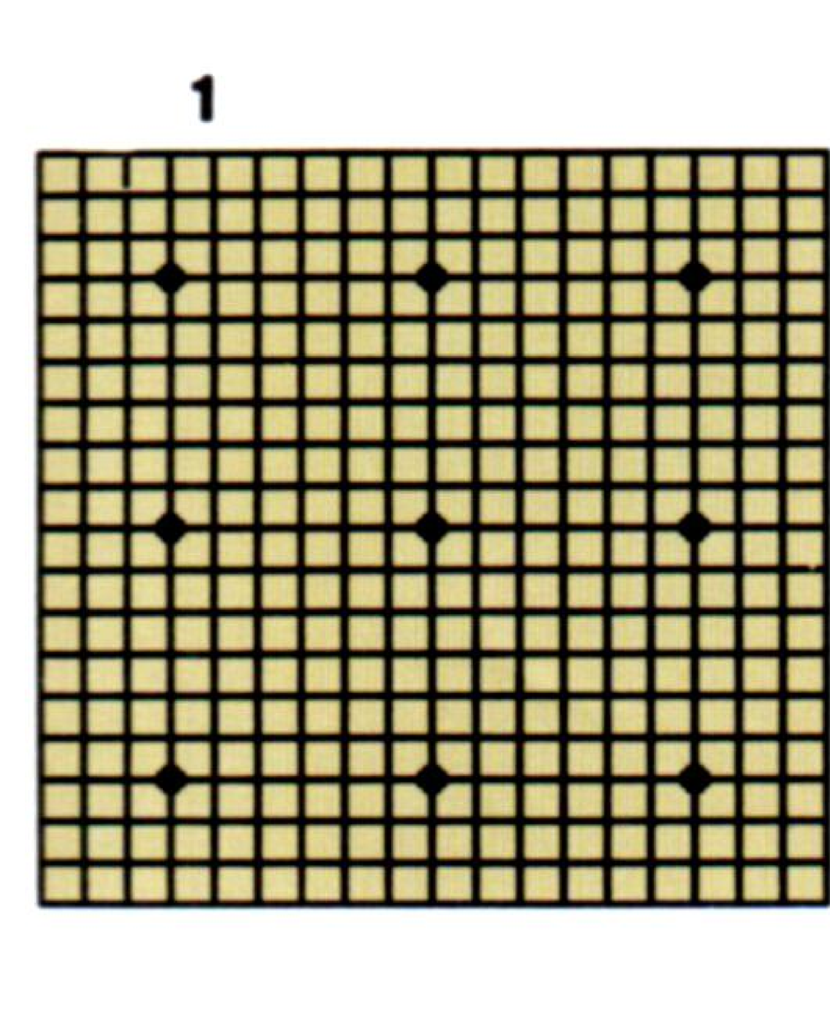
A lo largo de la serie desarrollaremos un programa para jugar al *go*. Aún no siendo sumamente eficaz, el programa proporciona una buena introducción al juego y se erigirá en un valioso oponente para el recién iniciado. El programa se ha diseñado específicamente para que sea fácil modificarlo y desarrollarlo.

El *go* es un juego para dos personas que se desarrolla en un tablero de 361 intersecciones (fig. 1). Los dos jugadores, que juegan el uno con fichas blancas y el otro con negras, colocan alternativamente una de ellas en cualquier intersección vacante del tablero. Observe que las fichas se colocan en las uniones de líneas y no en los cuadrados.

El objetivo del juego es rodear territorio (intersecciones vacantes) con las propias fichas de cada jugador. Para ganar, las fichas de uno deben hallarse rodeando el territorio más grande al final del juego.

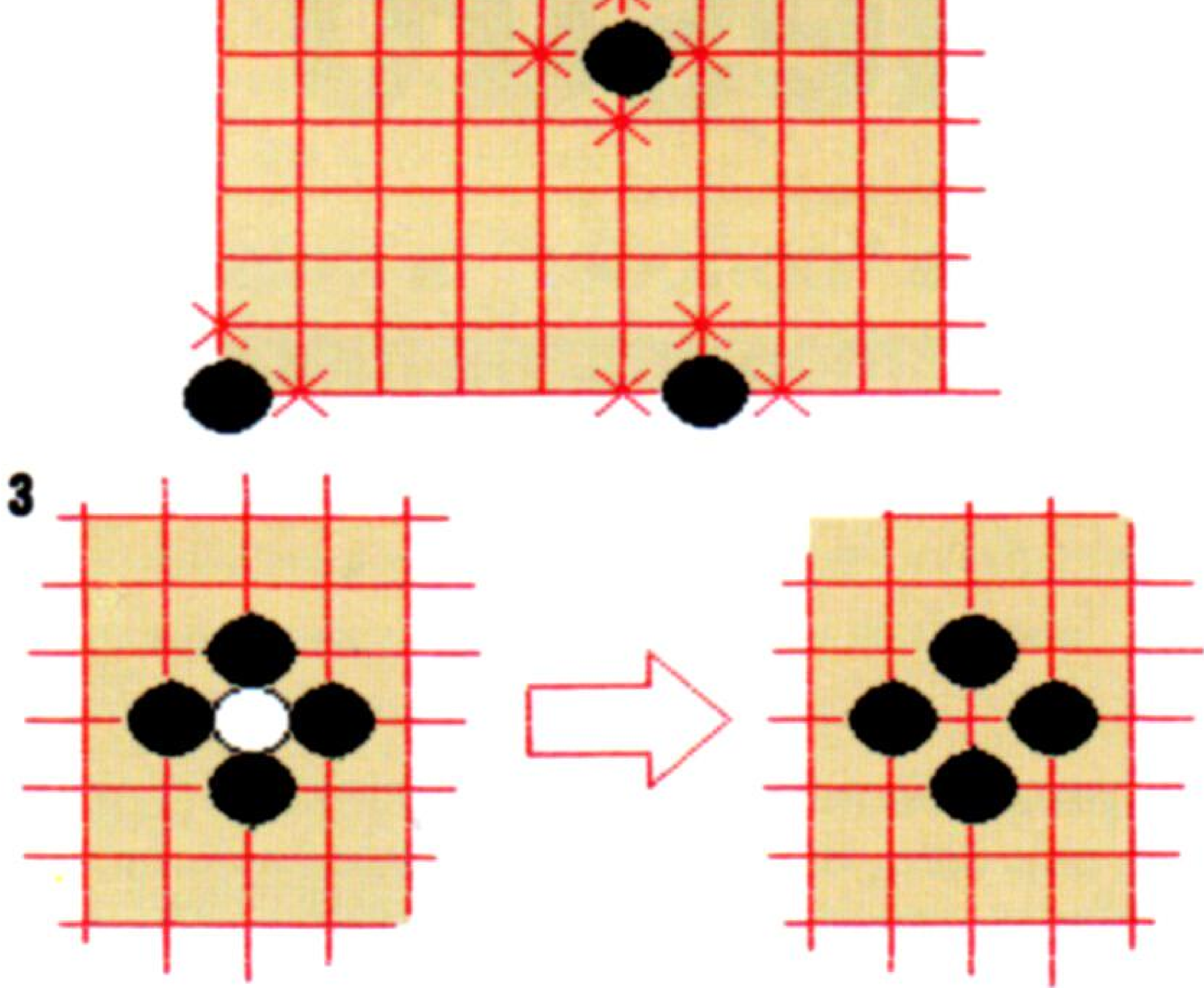
En *go* el primer movimiento siempre lo realizan las negras, que por lo general corresponden al jugador más débil. De haber una diferencia significativa entre las destrezas de juego de los dos jugadores, el primer movimiento de las negras consistirá en colocar entre dos y nueve fichas de *handicap* en el tablero, situándolas de acuerdo a un patrón específico en los nueve puntos marcados en el tablero (fig. 1).

1



Limitarse a rodear territorio es excesivamente fácil; también se pueden capturar fichas y eliminarlas del tablero. Todo punto vacante inmediatamente adyacente a una ficha (a lo largo de una línea) se denomina *licencia*. Por consiguiente, cada ficha individual puede tener dos, tres o cuatro licencias, según la posición que ocupe en el tablero (fig. 2). Para poder capturar una ficha del oponente, usted debe colocar su propia pieza en todas las licencias de la ficha del oponente. En la figura 3 vemos que, si el último movimiento de las negras fue ocupar la última licencia de la ficha blanca, esta última sería eliminada del tablero. Todas las fichas capturadas se suman al marcador del jugador.

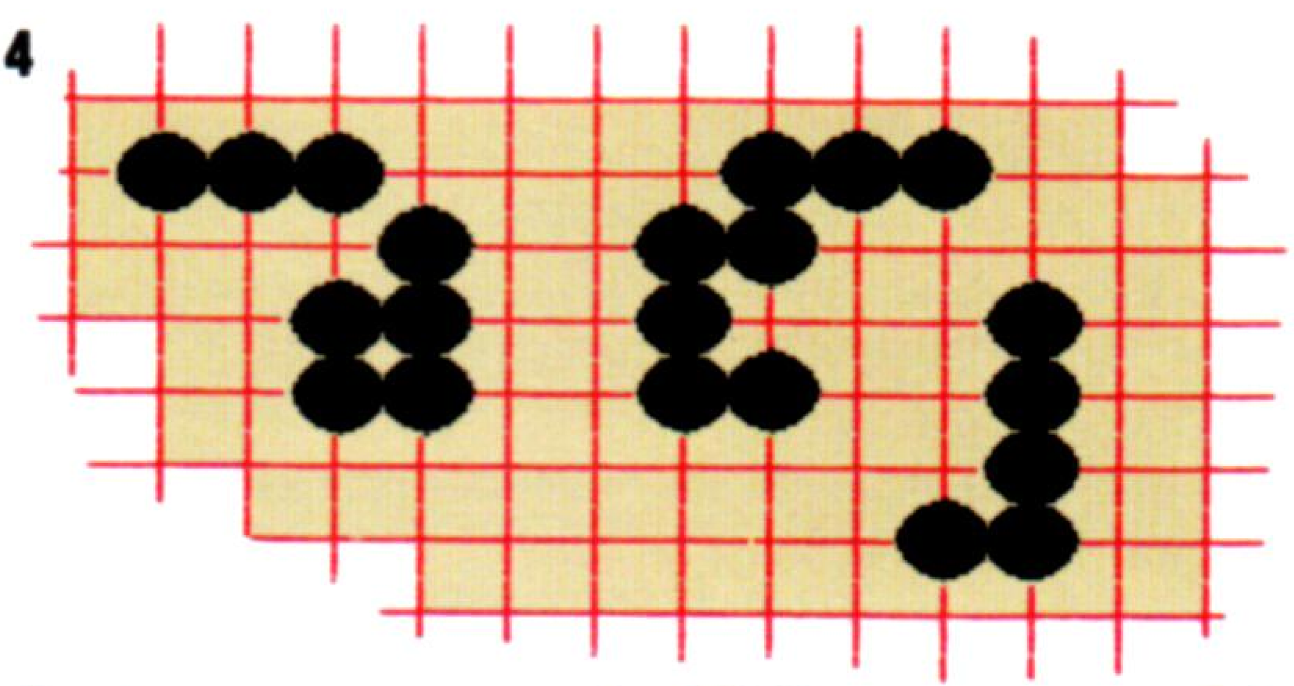
2



3

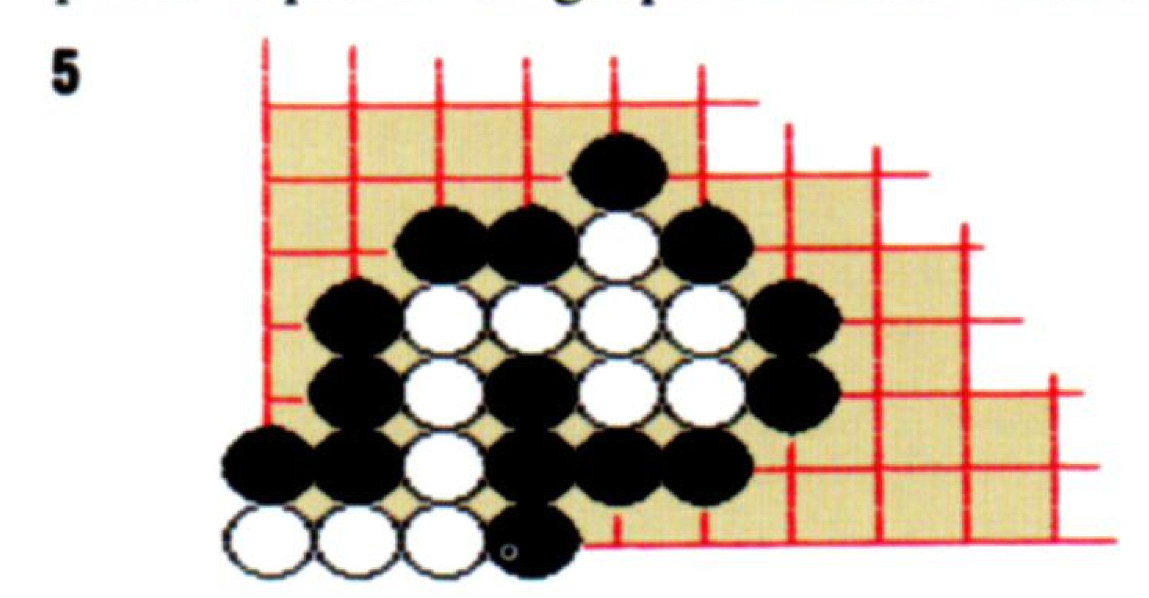
Si uno o más puntos adyacentes de una ficha se ocupan con otras fichas del mismo color, se dice que las mismas están conectadas y que forman un *grupo*. Las conexiones en diagonal no forman enlaces entre fichas: en la figura siguiente, por ejemplo, vemos cuatro grupos, y no tres:

4



Capturar grupos es más difícil que capturar fichas individuales, porque se han de eliminar como si se trataran de una sola unidad. Por tanto, las negras pueden tener que ocupar 15 puntos de licencia para poder capturar un grupo de fichas blancas.

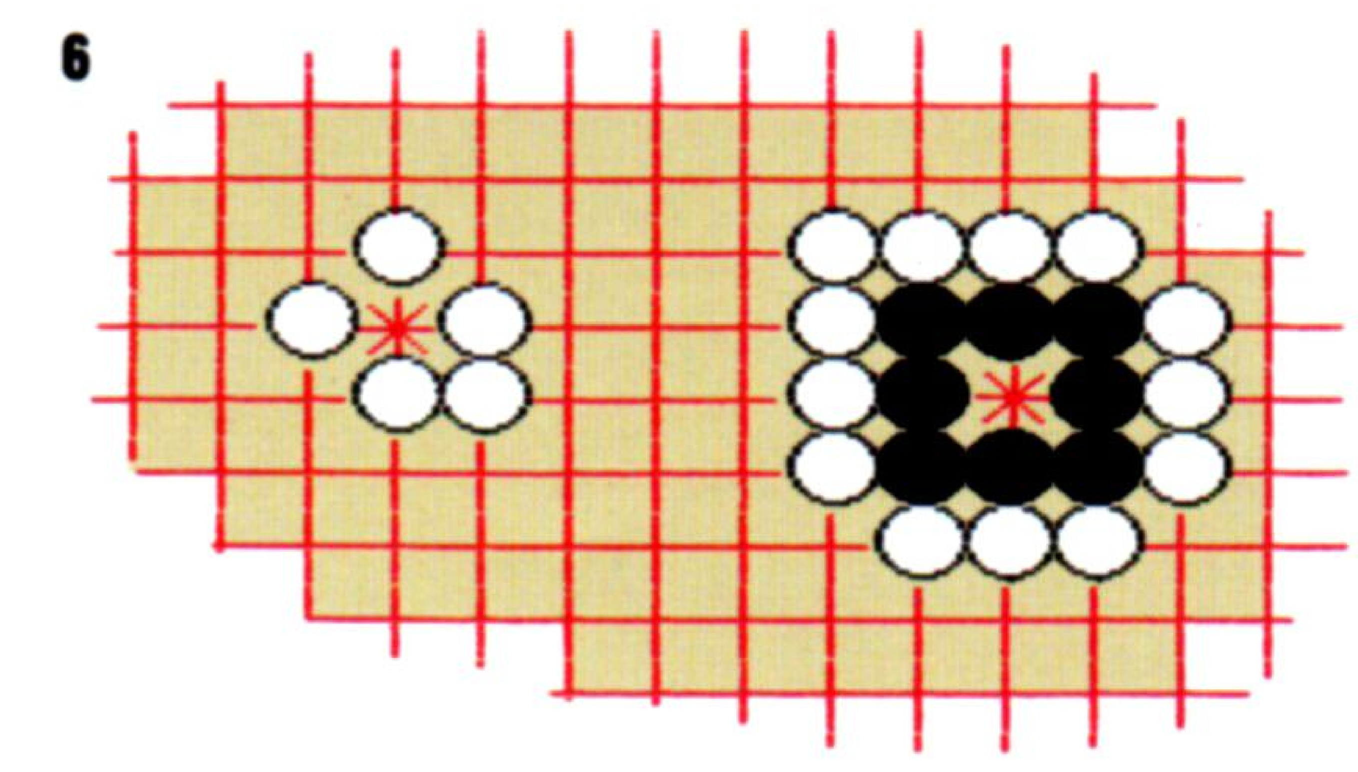
5

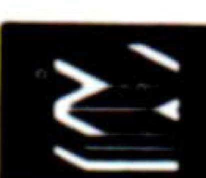


Esto en realidad es todo lo que se necesita saber para poder jugar al *go*, pero existen algunas complicaciones derivadas de las reglas reseñadas:

Suicidio: A tenor de las reglas, es posible que un jugador coloque una ficha de modo que no tenga ninguna licencia. Por ejemplo, si les tocara jugar a

6

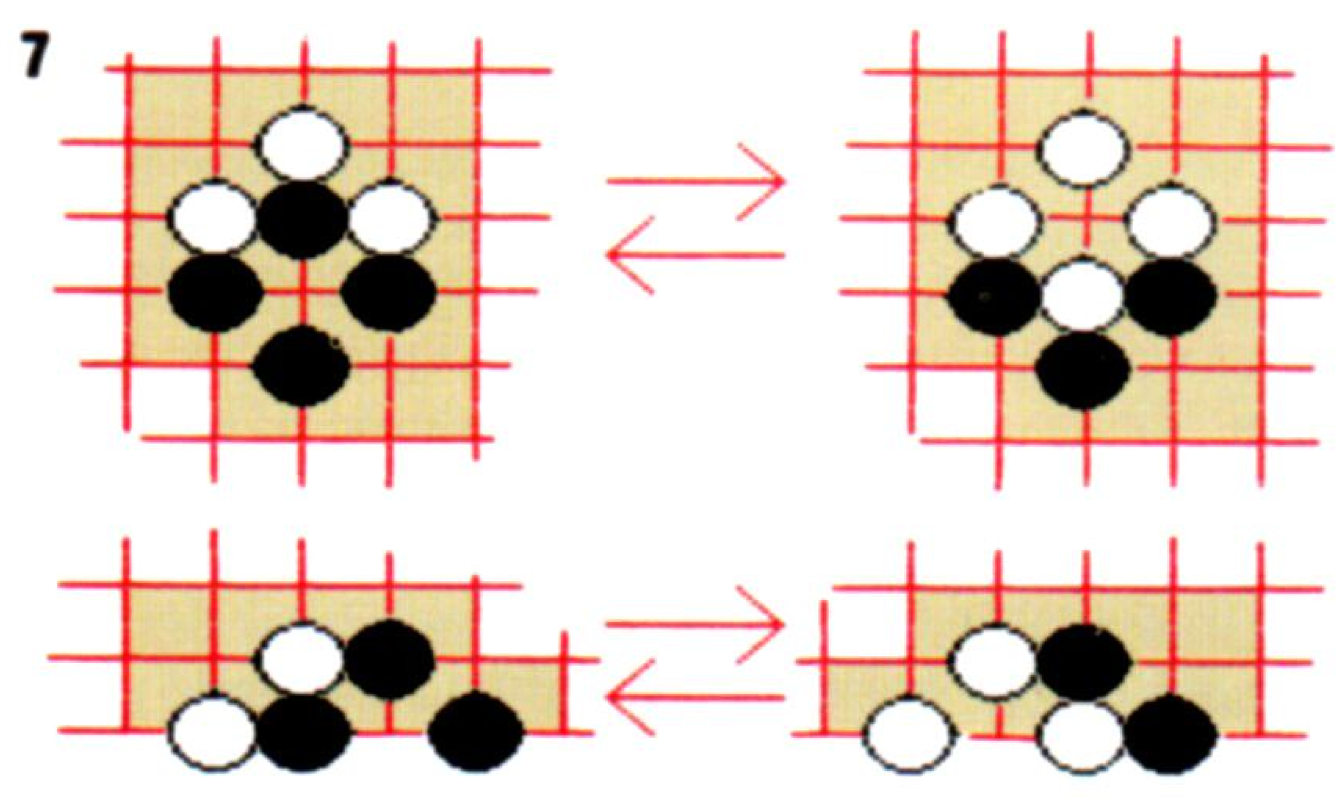




las negras, cualquiera de los puntos marcados en la figura 6 produciría esta situación. Este movimiento, denominado *suicidio*, está prohibido en el *go*.

Ko: Significa "infinitud" y alude a una posible situación en la que las fichas se podrían colocar y capturar indefinidamente. Los dos diagramas siguientes son casos comunes de esta situación.

7



Para impedir esta situación, las reglas del *go* prohíben que un jugador coloque una ficha de modo tal que se produzca la posición "inmediatamente anterior". Esta regla hace efectivas lo que se conoce como *luchas Ko*. Si no se permite que las blancas recapturen inmediatamente una ficha debido a la regla Ko, entonces, obviamente, se debe jugar una ficha intermedia. Ésta generalmente se coloca en una posición que amenace ya sea a una ficha negra o bien a un grupo, en algún otro lugar del tablero, y se conoce como un *Sente* (forzar una respuesta). Por tanto, las negras no pueden ocupar el punto Ko, que da por terminada la lucha Ko, y las blancas pueden recapturar la ficha Ko negra. Ahora las negras tienen prohibida la recaptura inmediata y deben tratar de hallar un movimiento Sente si es que quieren que la lucha Ko continúe.

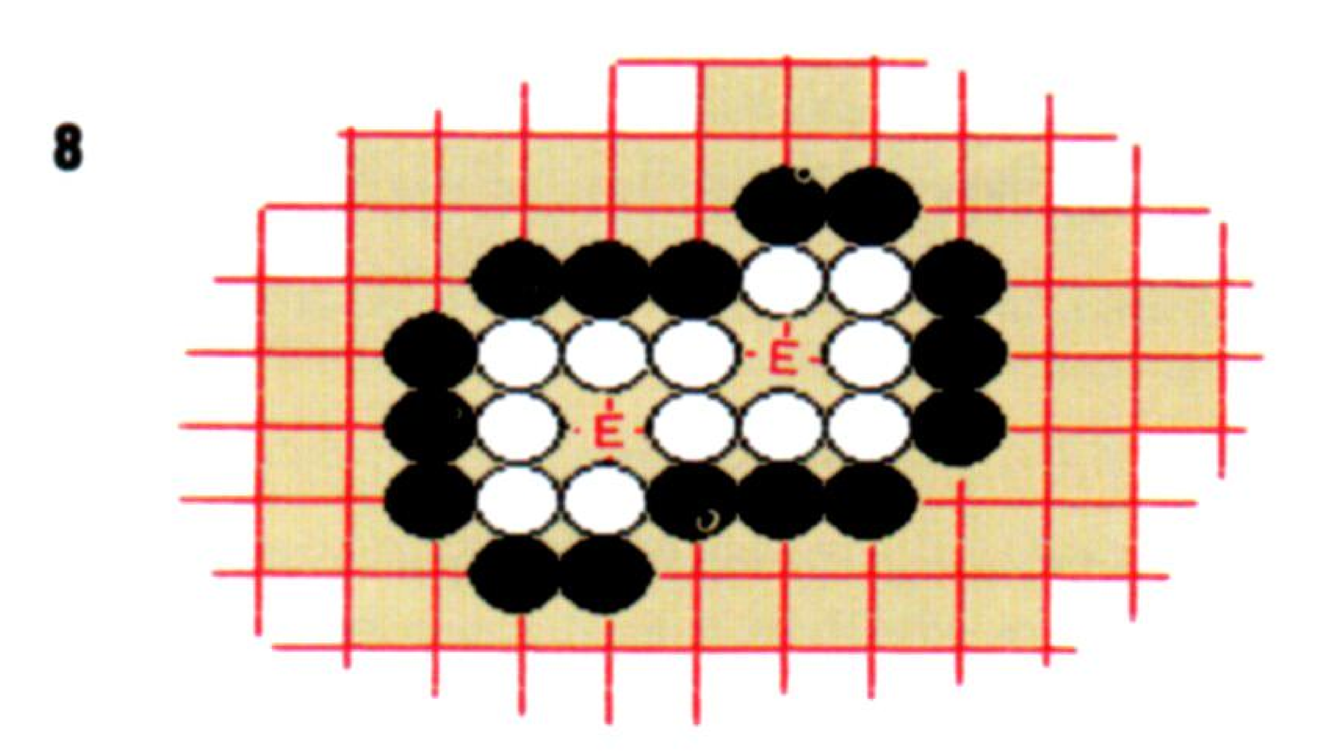
Un concepto muy importante que se desprende de estas reglas es el de *"vida y muerte"* para un grupo de fichas. Ya hemos visto que es imposible que las negras jueguen en la intersección vacante en el medio del grupo (fig. 6). Observe, sin embargo, que si aquí hubieran de jugar las blancas, el grupo negro sería capturado.

También sucedería que si el grupo negro no estuviera rodeado por fichas blancas, luego éstas no podrían colocar una ficha en este punto. Se desprende que si las blancas quisieran capturar este grupo negro, la última ficha blanca colocada debería hallarse en este punto vacante tras haber rodeado por completo el grupo negro con fichas blancas. El grupo negro sería, entonces, eliminado del tablero, dándole a la ficha blanca recién colocada cuatro puntos de licencia.

Se dice que el punto que se halla rodeado por fichas del mismo color de esta manera es un *ojo* y sucede que el ojo debe ser siempre la última licencia ocupada para capturar el grupo.

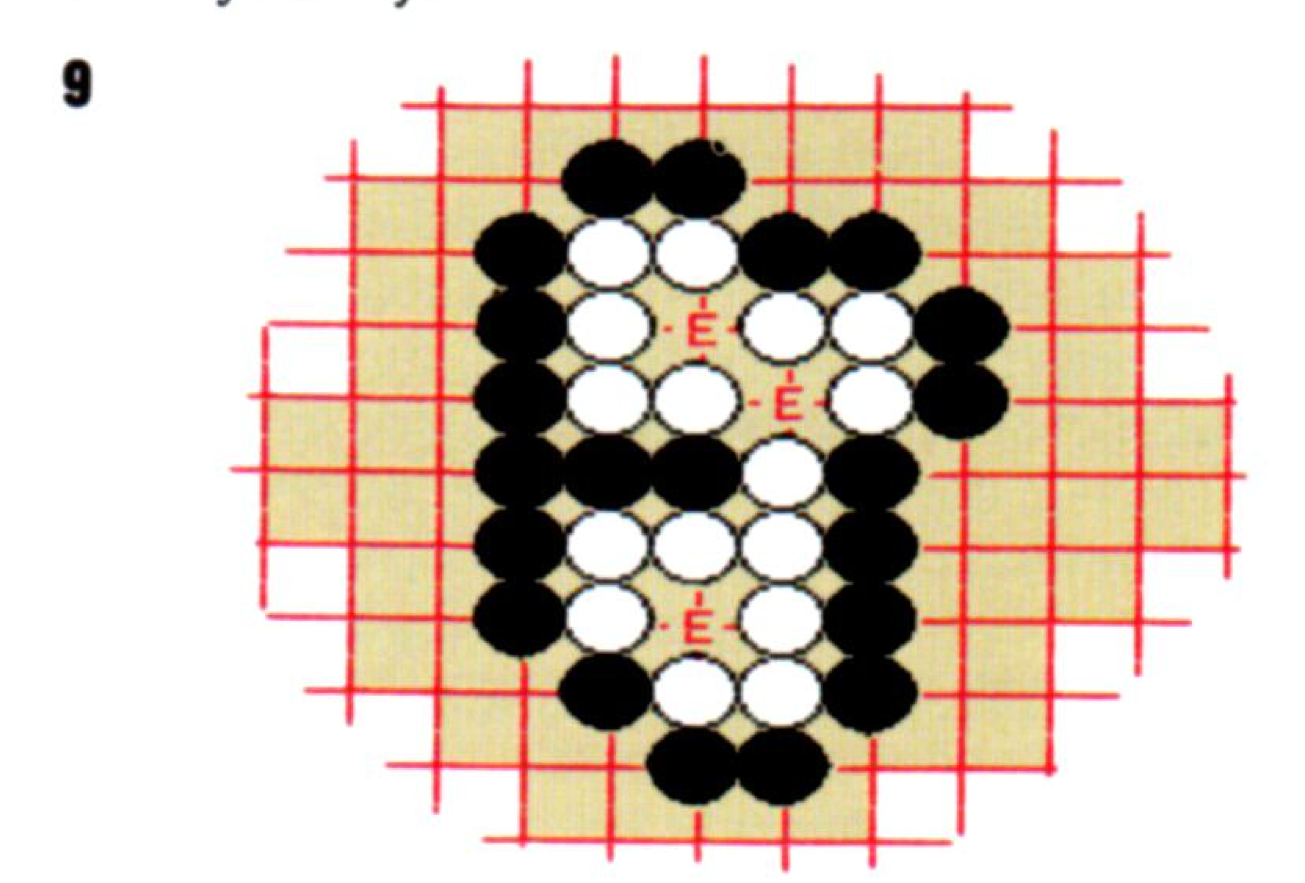
Avanzando un paso más este concepto, podemos imaginar un grupo que contenga dos ojos, como el grupo blanco que aparece en la figura 8. Para capturar al grupo, la última ficha negra debe ocupar ambos ojos simultáneamente; pero esto es imposible, ¡porque las negras sólo pueden colocar una ficha por vez! En consecuencia, este grupo (y todo el que contenga al menos dos ojos) se halla a salvo de captura, y permanecerá en el tablero hasta el final del juego (¡siempre y cuando, por supuesto, las blancas no incurran en el error de ocupar los ojos!).

8

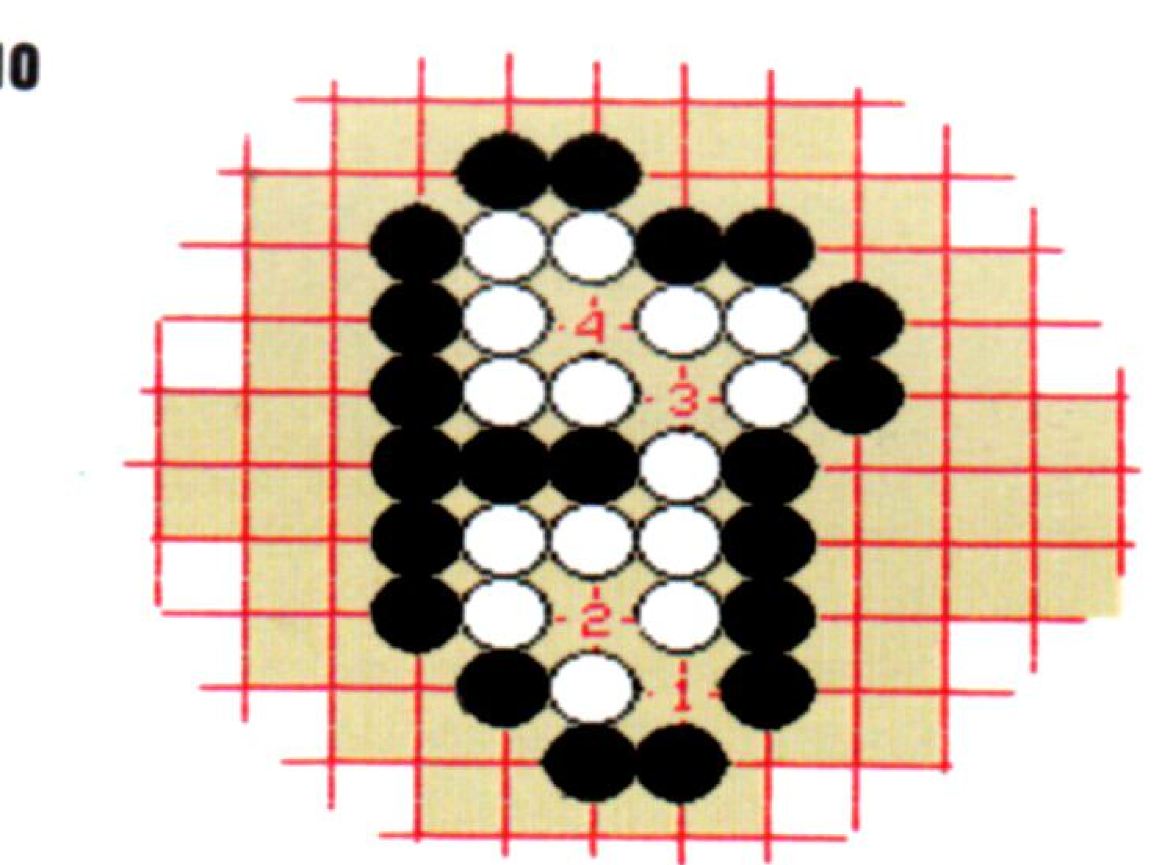


Ni siquiera es necesario que los ojos se hallen en un mismo grupo. Los tres grupos blancos de la figura 9 comparten tres ojos, y todas las fichas blancas están a salvo de ser capturadas. No obstante, usted debe tener cuidado. Mediante la secuencia indicada, las negras pueden capturar la formación similar de la figura 10 (que se diferencia en sólo una ficha). De los tres ojos, el de más abajo en realidad puede ser infiltrado por las negras; esto es lo que se conoce como *falso ojo*.

9



10



Todos estos grupos han tenido formaciones de ojos totalmente desarrollados, pero en la práctica no es necesario formar dos ojos para cada grupo, sino meramente contar con el potencial para hacerlo ante un eventual ataque.

El juego termina con el mutuo consentimiento de los jugadores, cuando ambos consideran que ninguna de las partes puede ganar ya nada más. Este final más bien ambiguo ha sido causante de muchos problemas en los programas de *go*, a los que aún les resulta muy difícil decidir cuándo debe terminar el juego. Éste también se da por terminado si ambos jugadores "pasan" en sus movimientos, devolviéndole el control al oponente. Durante el juego se les permite a ambos jugadores pasar en cualquier momento, pero ello se hace sólo muy raramente.

¿Quién gana?
Al final del juego, se calcula así el marcador de cada jugador:

1. Se ocupan todos los puntos neutrales, denominados *damas*. Éstos los puede ocupar cualquiera de los jugadores, ya que no contarán para el marcador final.

2. Todas las fichas o grupos que no puedan evitar la captura se eliminan del tablero, como si hubieran sido capturados. Ello implica que no es necesario capturar estas fichas "muertas" durante el juego, si bien usted puede hacerlo si así lo desea. Se podría efectuar la secuencia de captura, pero no beneficiaría a ninguno de los jugadores. (Cada ficha defensiva colocada incrementaría el marcador del jugador atacante en un punto, pero este jugador también tendría que colocar una ficha, ocupando por tanto su territorio, y reduciendo su marcador en un punto.)

3. Luego se calcula el marcador de cada jugador como el número de intersecciones vacantes (el territorio) controlado por él, menos el número de fichas capturadas por el oponente. Gana quien obtiene más puntos.

Líneas melódicas

Nos hallamos en la segunda fase del proyecto: la conexión de las patillas del chip a las líneas de datos y de control del ordenador

El chip adaptador de interface para comunicaciones asíncronas (ACIA) constituye la base de la interface MIDI y es compatible con los procesadores 6502 y 6510 que utilizan el BBC Micro y el Commodore 64, respectivamente. Por lo tanto, podemos enlazar las líneas de datos de dirección y de control del procesador con las patillas del chip.

La placa final está diseñada para enchufarse directamente en la puerta para ampliación del Commodore 64. La versión para el BBC Micro debe conectarse en la puerta de tubo de la cara inferior del ordenador mediante un cable plano de 40 vías y conectores adecuados. Explicaremos por separado las conexiones que se deben realizar para cada micro. Una vez construida la placa, se pueden llevar a cabo algunas pruebas simples para asegurar que funcione correctamente. Estas pruebas implican colocar datos en los registros del chip ACIA y efectuar un simple bucle de comprobación.

Es necesaria la decodificación de direcciones para acceder a cualquier periférico que esté conectado al bus de datos principal, pero que no posea la misma cantidad de líneas de dirección que la CPU. Tales dispositivos suelen poseer una patilla *chip select* que permite que el dispositivo utilice el bus de datos cuando la línea de selección de chip está activa. Las líneas de conexión conectadas al dispositivo permitirán la selección de registros diferentes.

Las señales de selección de chip para los diversos dispositivos conectados al sistema se generan decodificando las líneas de dirección más significativas. Se puede utilizar cada combinación de bits de estas líneas de dirección para seleccionar exclusivamente un dispositivo. De este modo, las registros internos del dispositivo aparecerán en el mapa de memoria de la CPU, permitiendo acceder a ellos como si fueran posiciones normales de memoria. Sin embargo, se debe tener cuidado en asegurar que las direcciones utilizadas por el dispositivo no correspondan a registros relevantes empleados por el OS del ordenador anfitrión.

La puerta para ampliación del Commodore 64 tiene dos salidas, etiquetadas I/01 e I/02. Estas líneas se ponen *low* (bajo) cuando se accede a las páginas $DE y $DF, respectivamente. Conectando simplemente la línea CS2 del chip ACIA con I/01, podemos asociar el chip a la página $DE. Debido a que el ACIA no está conectado a las líneas de dirección de la A1 a la A7, se puede acceder a los registros internos del chip mediante cualquier dirección comprendida entre $DE00 y $DEFF. Conectando A0 directamente a la patilla de selección de registro del ACIA, todas las direcciones pares accederán a los registros de datos de transmisión/recepción. La elección más obvia para las direcciones es $DE00 (decimal 56832) y $DE01 (decimal 56833).

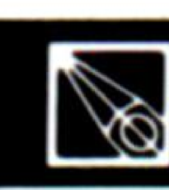
El BBC Micro Modelo B ofrece dos posibles puertas que se pueden utilizar para conectar la interface, cada una de las cuales presenta sus ventajas y sus inconvenientes. La puerta de tubo nos permite el acceso directo al bus de datos de 2 MHz. Se proporciona una línea de decodificación de dirección, NTUBE, para las direcciones desde &FEE0 hasta &FEFF. El inconveniente es que el BBC Micro comprueba si hay presente algún dispositivo en el tubo mediante la lectura de ciertas direcciones del tubo cuando éste se enciende. Si hubiera instalado algún otro dispositivo distinto del segundo procesador, el ordenador parecerá colgado mientras espera datos provenientes del segundo procesador.

Una respuesta profesional a este problema sería conectar una de las líneas de dirección de mayor orden al CS0 en el chip. Una alternativa es conectar NTUBE a CS2 y enchufar la placa en la puerta de tubo con el ordenador encendido. Por razones de simplicidad, ésta es la solución que hemos elegido para este proyecto. Sin embargo, otra alternativa es usar el bus de 1 MHz. Las patillas marcadas NPGFC y NPGFD se proporcionan para asociar dispositivos externos en las páginas &FC y &FD y se podrían utilizar de la misma forma en que se utilizan I/01 y I/02 en la versión para el Commodore 64.

La utilización del bus de 1 MHz tiene dos desventajas: primero, debido a que el reloj del sistema trabaja más despacio, a 1 MHz, en las dos líneas de decodificación se producen "desperfectos". Para evitar este problema debe utilizarse un circuito de limpieza (como el que se describe en la sección 28.5 de la guía para el usuario avanzado del BBC Micro). El segundo problema es que no hay ninguna alimentación de potencia de 5 V al bus de 1 MHz. En consecuencia, la potencia debe derivarse ya sea desde la puerta para el usuario o bien desde el conector auxiliar de potencia.

Un poco de sabiduría

Si desea programar la interface MIDI por sí mismo, es importante que comprenda las funciones de los cuatro registros del chip ACIA. Éste se puede programar a través del registro de control para interrumpir a la CPU cuando se establezcan ciertos bits de estado en las secciones de recepción y/o transmisión del ACIA. Se producirá una interrupción del transmisor si los bits 5 y 6 del registro de control están a 1 y a 0, respectivamente, y el bit de estado TDR está establecido. La interrupción se elimina escribiendo datos en el TDR. Se produce una interrupción del receptor si tanto el bit de control 7 como el bit de estado 0 están a 1. La lectura del RDR eliminará la interrupción (a menos que también esté a 1 el bit de estado 5, que se elimina leyendo el registro de estado antes que el RDR). El bit de estado 2 también puede generar una interrupción, pero ésta



Los registros del ACIA

	BITS DE DATOS	BITS DE FINAL	PARIDAD
000	7	2	PAR (*)
001	7	2	IMPAR (*)
010	7	1	PAR (*)
011	7	1	IMPAR (*)
100	8	2	NINGUNA (*)
101	8	1	NINGUNA (*)
110	8	1	PAR (*)
111	8	1	IMPAR (*)

no es aplicable aquí. Se proporcionan líneas separadas para los relojes de recepción y transmisión, pero suelen estar conectadas juntas. El reloj de control de velocidad en baudios deriva directamente de la entrada del reloj o bien dividiendo por 16 o 64, según el estado de los bits de control 0 y 1. Los bits 2, 3 y 4 del reg. de control determinan la cantidad de bits de final y de bits de datos y si la paridad es par, impar o bien no se usa. Para la MIDI necesitamos los bits 2 y 4 en 1 y al bit 3 a 0 (ocho bits de datos, ninguna paridad, un bit de final) para adecuarnos al estándar. Los bits 5 y 6 del reg. de control gestionan el control del transmisor. Para la MIDI, necesitamos que el bit 6 esté a 0 y entonces el bit 5 activará las interrupciones del transmisor

Commodore 64

CARA DE LOS COMPONENTES

+5 V

0 V

22 21 20 19 18 17 16 15

D7–D0

9 CS2 13 R/W 7 IRQ

CARA DE COBRE

ATAR A 0 V EN CARA DE LOS COMPONENTES

A0 11

0 V

Ø 14

BBC Micro

CARA DE LOS COMPONENTES

A0 11

15 16 17 18 19 20 21 22

D0–D7

CS2 9 IRQ 7 Ø 14 13 R/W

CARA DE COBRE

+5V

0V

Conexiones marginales

Estos diagramas muestran las conexiones apropiadas a las patillas del chip ACIA para cada micro. Los cables desde los conectores marginales están numerados. Éstos están relacionados con los números de patillas del chip ACIA. Observe que las patillas de este chip están numeradas del siguiente modo: sosteniendo el chip con la muesca situada arriba de todo y las patillas alejadas de usted, la patilla número 1 es la más próxima a la muesca del lado izquierdo. Las patillas se numeran luego hacia abajo por el lado izquierdo, hasta el número 12. La patilla 13 es la opuesta a la patilla 12 sobre el lado derecho, y las restantes patillas se numeran yendo hacia arriba por el lado derecho hasta la patilla 24, la más próxima a la muesca. Emplee alambre de enlace para realizar las conexiones apropiadas.
Una vez terminada, debe insertarse la versión para el Commodore 64, con el lado de los componentes hacia arriba, en la puerta para ampliación del 64, y encender el ordenador, listo ya para las pruebas. Para la versión del BBC Micro, es necesario confeccionar un cable de conexión usando un metro de cable plano de 40 vías, un conector de patillas IDC de 40 vías y un conector marginal de 20 vías

Kevin Jones



Comprobación de la placa

Con la placa ya enchufada, podemos llevar a cabo algunas pruebas sencillas para verificar el correcto funcionamiento de la placa. Será útil un tester sencillo para aislar cualquier fallo, en la improbable circunstancia de que la placa fracase en alguna de las pruebas. Si no tiene un tester, necesitará llevar a cabo una concienzuda inspección visual de la placa.

1. Si el ordenador no funciona normalmente con la placa insertada, compruebe lo siguiente:

- Verifique que el voltaje entre las líneas de +5 V y 0 V sea realmente de 5 V. De no ser así, compruebe que todos los IC estén insertados correctamente. Examine la placa por si hubiera cortocircuitos entre las pistas de potencia.
- Quite la placa del ordenador y use el tester para comprobar la existencia de cortocircuitos entre cualquiera de las conexiones del bus del ordenador.

2. Cuando el ordenador parezca funcionar normalmente, conecte un cable MIDI entre los conectores MIDI IN y MIDI OUT utilizando cables conectores DIN de *hi-fi* normales de 5 patillas. Digite la siguiente instrucción (entre paréntesis, el equivalente para el BBC Micro):

```
POKE 56832,3   (?&FEE0=3)
```

Esta instrucción coloca un 3 en el registro de control ACIA, que realiza una reinicialización maestra. Ahora digite:

```
POKE 56832,22   (?&FEE0=&16)
```

Ésta coloca el valor $16 en el registro de control y configura al ACIA del siguiente modo:

- Inhabilita las interrupciones del receptor y el transmisor (porque aún no disponemos de medios para manipularlas).
- Define las palabras en serie transmitidas y recibidas como ocho bits de datos más un bit de final sin ninguna generación/comprobación de paridad.
- Define la velocidad en baudios como el reloj en las patillas 3 y 4 dividido por 64. (2 MHz/64=31.25 KHz, que es la velocidad especificada para la MIDI).

Ahora el ACIA debe estar listo para recibir y transmitir datos. Compruébelo leyendo el registro de estado mediante:

```
PRINT PEEK(56832)
(PRINT?&FEE0)
```

Usted deberá leer el valor 2. Éste indica que tanto el registro de datos a transmitir (bit 1 a 1) como el registro de datos a recibir (bit 0 a 0) están vacíos. Puesto que no se han recibido datos y las interrupciones están desactivadas, los bits de estado restantes, del 2 al 7, deben estar a 0.

3. Envíe un byte desde el registro de transmisión a través del cable hacia el registro de recepción, mediante:

```
POKE 56833,X   (?&FEE1=X)
```

donde X es cualquier número entre cero y 255. Esta instrucción coloca un valor en el registro de datos, para transmitirlo.

4. En una fracción del tiempo que lleva digitar la siguiente instrucción, se debe haber recibido el byte. Vuelva a leer el registro de estado:

```
PRINT PEEK(56832)
(PRINT?&FEE0)
```

Ahora el valor deberá ser 3. El bit 1 se habrá borrado inmediatamente después de la última instrucción (registro de transmisión completo), pero se establecerá a 1 al cabo de un corto tiempo, apenas el registro de datos a transmitir esté listo para el siguiente byte. El bit 0 estará establecido a 1, indicando que el byte se ha recibido y que se lo puede leer desde el registro de datos. Observe que hay un posible error: si el bit 0 no está a 1, probablemente haya un circuito abierto en el camino de transmisión que esté manteniendo la entrada del receptor a un nivel *high* (alto), impidiendo su detección.

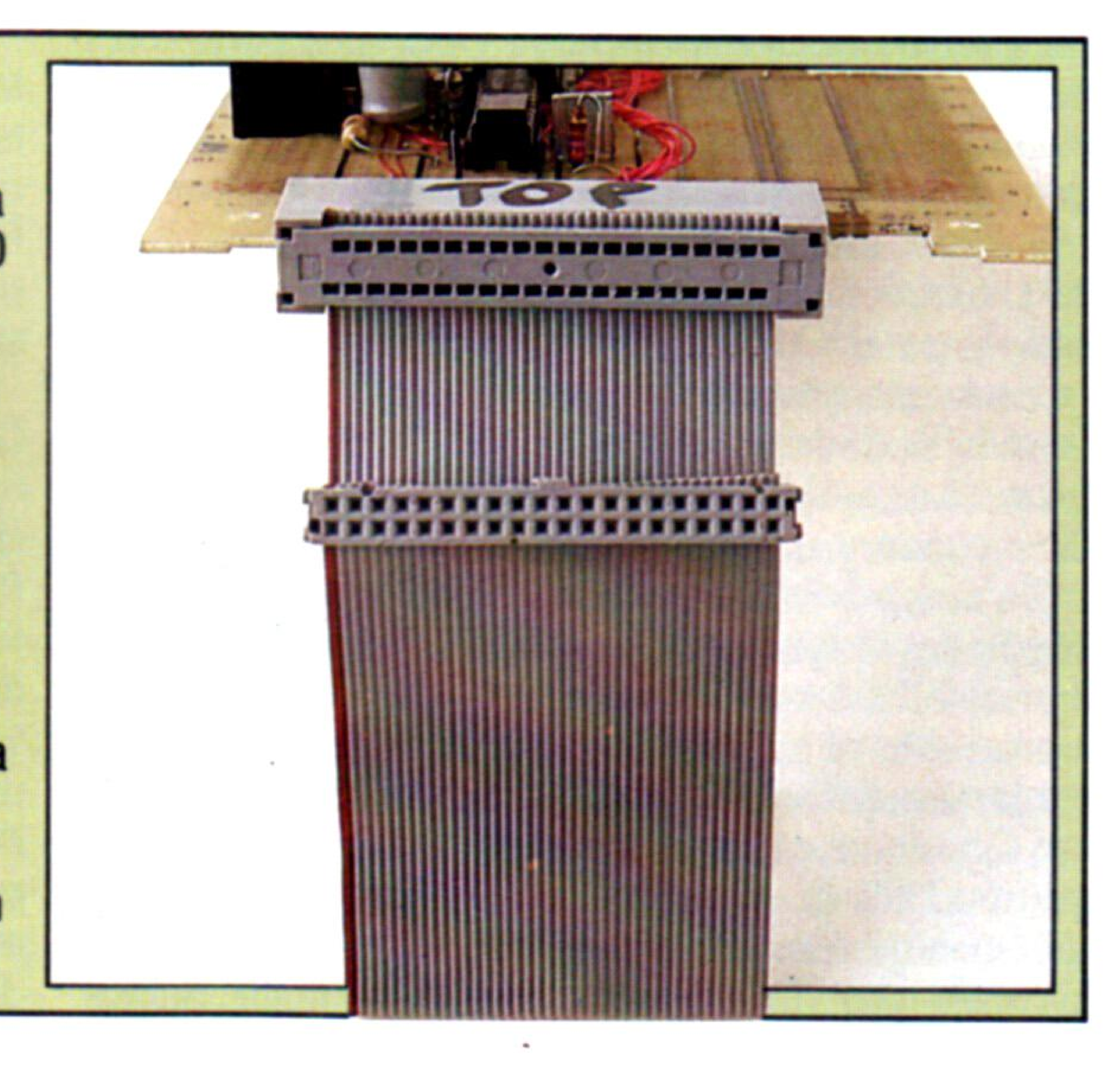
5. Habiendo verificado que se haya recibido un byte, podemos comprobar si es el mismo que el transmitido leyendo el registro de datos:

```
PRINT PEEK(56833)
(PRINT ? &FEE1)
```

Esta operación de lectura debe devolver el mismo valor X que se transmitió antes. Los pasos 3, 4 y 5 deben repetirse varias veces con distintos valores de X

Desarrollo plano

La placa de la interface MIDI se puede unir a la puerta de tubo del BBC Micro mediante un cable plano de 40 vías, un conector marginal IDC de 40 vías que se instala en el borde de la placa y un conector de patillas IDC de 40 vías. Este cable se enchufa directamente en la puerta de tubo, en la cara inferior del BBC Micro. Al montar cable y conectores, observe la orientación de los dos enchufes. Estire el cable de modo que quede plano y fije el conector marginal de modo que la patilla 1 (marcada en la carcasa del conector) quede en la fila de abajo. Haga una marca en la superficie superior de la carcasa para señalar la orientación del conector cuando sea utilizado con la placa de la interface. El conector de patillas IDC se debe fijar al otro extremo del cable con la muesca rectangular arriba (véase fotografía)



Un agente veloz

Con el Phonemark 8500 Quick Data Drive, que utiliza "wafers" de floppies serializados, los usuarios de Commodore mejoran sus condiciones de almacenamiento externo



Chris Stevens

Estimable alternativa
A pesar de las críticas de que han sido objeto el aparato de cassette y la unidad de disco de Commodore, ha habido pocos proveedores "independientes" de sistemas de almacenamiento alternativos para estas máquinas. La Quick Data Drive es un sistema de wafer flexible serializado que utiliza bucles de cinta continuos idénticos a los de la Rotronics Wafadrive. El sistema no es particularmente barato, pero vale la mitad que la unidad de disco Commodore y en muchas aplicaciones es considerablemente más rápido

Haciéndose eco de la demanda de muchos entusiastas de los ordenadores personales, que desean contar con métodos más rápidos y eficaces para acceder a sus programas, los fabricantes están produciendo una amplia gama de dispositivos de almacenamiento masivo dirigidos a los micros más populares. Entre estos dispositivos se halla el Phonemark 8500 Quick Data Drive, de Dean Electronics, fabricado para el Commodore 64 y el Vic-20.

Este dispositivo es un pariente cercano de la Rotronics Wafadrive, diseñada para el Sinclair Spectrum. Las unidades para ambos las fabricó BSR Electronics y utilizan *wafers* idénticos.

La unidad de disco estándar Commodore 1541 es conocida, al igual que la unidad de cassette, por su lentitud. Ello no se debe a la unidad de disco en sí misma sino al método que utiliza el ordenador para cargar los datos. Gran parte del sistema operativo del Commodore 64 es herencia de las máquinas de gestión PET de mediados de los setenta, cuando los sistemas de almacenamiento masivo, en particular las unidades de cassette, eran muy poco fiables como dispositivos para almacenamiento de datos. Cuando Commodore concentró su atención en proporcionar almacenamiento de apoyo para la serie PET, decidió suministrar su propia unidad de cassette e incorporar una serie de comprobaciones para asegurar que los datos cargados fueran correctos. Si bien ello incrementó la fiabilidad del proceso de carga (LOAD), lo logró a costa de la velocidad de acceso. Este sistema se traspasó al Vic-20 y posteriormente al Commodore 64.

En la actualidad, la calidad de la cinta de cassette ha mejorado mucho y la necesidad de una comprobación de datos larga y complicada para las máquinas Commodore ha desaparecido. Muchos paquetes de software comerciales contienen actualmente sus propias técnicas de carga, que eliminan muchas de las comprobaciones y hacen que la carga se realice con mayor rapidez, sin pérdida de fiabilidad.

No obstante, al cargar sus propios programas, la mayoría de los usuarios no tienen acceso a estas técnicas de gran velocidad y deben aún padecer las demoras impuestas por el sistema operativo Commodore. Se puede considerar que la Quick Data Drive, de la cual se afirma que carga 15 veces más rápido que las cassettes normales y más rápido que la unidad de disco 1541, representa una alternativa comercial a los periféricos Commodore.

La Quick Data Drive es un dispositivo bastante pequeño, de poco más de la mitad del tamaño del aparato de cassette Commodore 1530. La unidad se conecta al Vic-20 o al Commodore 64 a través del conector marginal para cassette. A diferencia de la Rotronics Wafadrive, la Quick Data Drive posee sólo una unidad. Aunque sería preferible contar con dos unidades cuando se necesita operar simultáneamente un wafer de sistema y un wafer de datos, la unidad única resulta adecuada a la mayoría de los usuarios. Si fuera absolutamente necesario, siempre se podrá adquirir una segunda unidad y conectarla para proporcionar una capacidad de unidad dual.

El hecho de que la unidad se enchufe en la puerta para cassette no significa que usted no pueda tener en funcionamiento, al mismo tiempo, un aparato de cassette. En la parte posterior de la Data Drive hay un conector marginal para cassette que permite encadenar en margarita unidades de cassette o una segunda Quick Data Drive.

Si bien la Rotronics Wafadrive y la Quick Data Drive poseen un aspecto similar, sus métodos de operación son muy diferentes y reflejan las diferencias existentes entre los ordenadores para los que fueron diseñadas. Mientras que el sistema operativo de la Wafadrive está retenido en ROM (imitando a las ROM de la Interface 1), el sistema operativo de la Quick Data Drive (QOS) está retenido en un wafer. Para cargar (LOAD) el QOS en el ordenador, se debe pulsar Shift/Run (tal como se haría para cargar una cassette normal). Cuando aparece en pantalla la indicación PRESS PLAY ON TAPE, se debe pulsar un pequeño botón situado en la parte

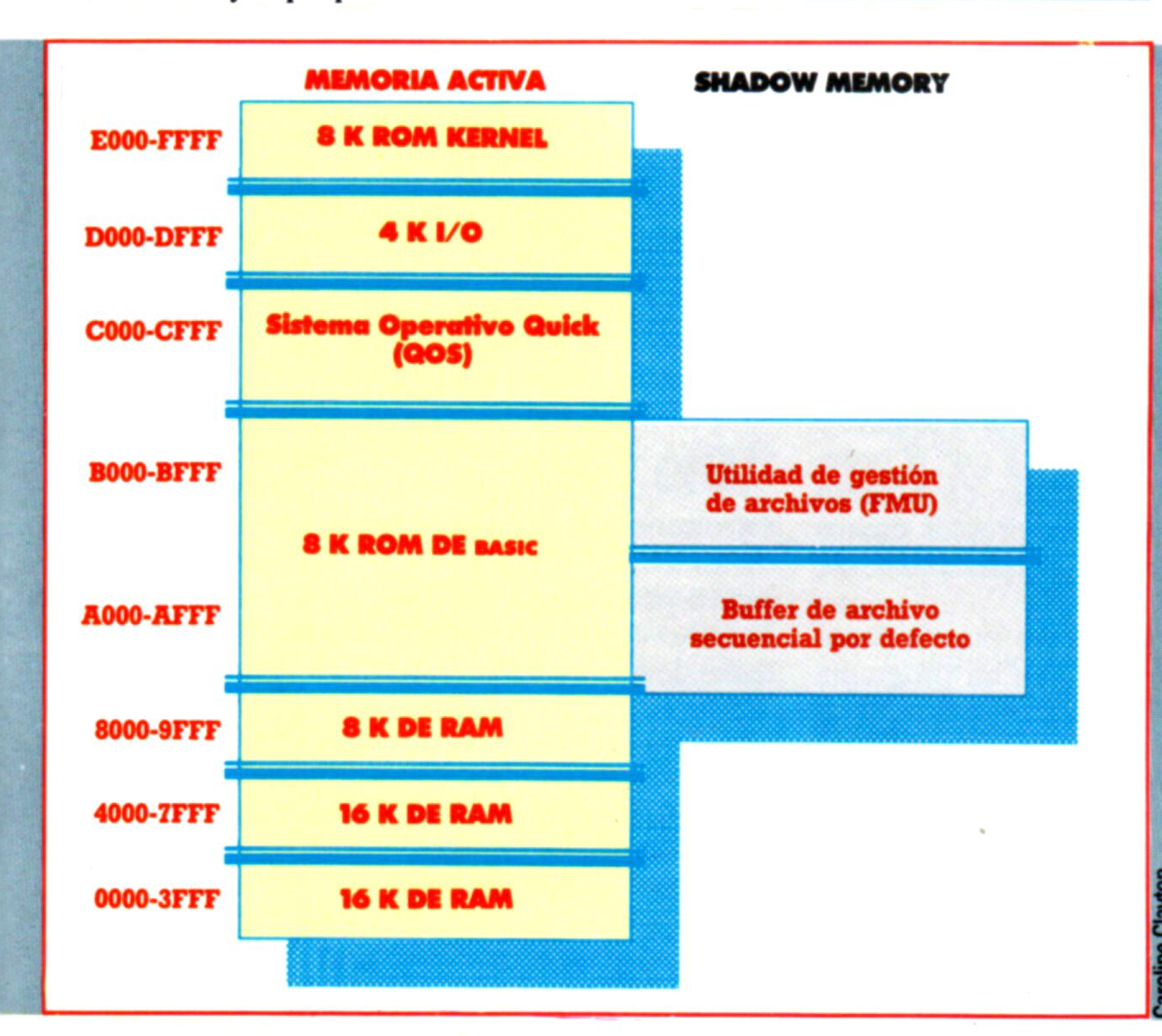
posterior de la unidad, que carga automáticamente el sistema. Una vez hecho esto, el QOS lo llevará a cabo de forma automática para los subsiguientes accesos.

Los programas que componen el QOS se cargan en dos zonas separadas de la memoria. Primero se almacenan las rutinas en código máquina para cargar (LOAD), guardar (SAVE) y buscar programas, entre las direcciones C000 y CFFF hacia la parte superior de la memoria (normalmente utilizada para programas en código máquina). Es este módulo del QOS el responsable de la mayor velocidad de carga de la unidad.

Aunque el QOS no implementa ninguna instrucción propia (al utilizar aquellas ya disponibles en la ROM del sistema operativo Commodore), intercepta las rutinas que se ocupan de la carga normal e inserta las suyas propias.

QUICK DATA DRIVE

DIMENSIONES	147×118×49 mm
INTERFACES	Conector para la puerta de cassette del Commodore 64 y el Vic-20
FORMATO	Wafers flexibles serializados de bucle continuo
CAPACIDAD	A la venta wafers de 16, 64 y 128 Kbytes
VELOCIDAD	Tiempo de acceso promedio: 8 seg/archivo de 15 Kbytes; 43 seg/archivo de 120 Kbytes



Una rápida memoria
La Quick Data Drive no conserva su sistema operativo en ROM sino que ha de cargarlo desde un wafer de sistema. Los diversos componentes se instalan en dos zonas separadas de la memoria. El primero, el Sistema Operativo Quick (QOS: *quick operating system*), se carga en la zona normalmente reservada a programas en código máquina. La otra parte del sistema, la utilidad de gestión de archivos (FMU: *file management utility*) se retiene en la "RAM en sombras", detrás de la ROM de BASIC. Aquí también se halla contenido un buffer de archivos en el cual se cargan los programas antes de guardarlos en otro soporte

La otra parte del Quick Operating System (QOS) es la FMU (*file management utility*: utilidad para gestión de archivos), que contiene numerosas rutinas útiles. Ésta se mantiene en la mitad superior de los 8 K de la Shadow Memory por debajo de la ROM de BASIC, entre las direcciones B000 y AFFF. Los 4 K inferiores entre A000 y AFFF se utilizan como un tampón o buffer de archivos secuenciales que emplea la FMU. Debido a que la FMU está almacenada en la zona de memoria por debajo de la ROM de BASIC, no se pueden utilizar ambas al mismo tiempo. Para llamarla desde la memoria debe ejecutarse la instrucción LOAD 'FMU'. Ésta simplemente "blanquea" la ROM de BASIC y da entrada a la FMU.

Sin ninguna duda, las operaciones de la Quick Data Drive son mucho más rápidas que los métodos de cassette estándares. A modo de ejemplo, se utilizó como programa de comprobación el juego de simulación *El Nuevo Mundo*, desarrollado en nuestro apartado "Programación". Cargar el programa completo de 25 K, lo que consumía más de nueve minutos desde cassette y un minuto y medio desde disco, llevó apenas 30 segundos con la Quick Data Drive, lo que representa una mejora considerable. No obstante, coomo con todos los sistemas basados en bucle de cinta, ello depende en gran medida de dónde se halle la cabeza de lectura/escritura de la unidad en relación al comienzo del programa.

El Sistema Operativo Quick encuentra los archivos de datos requeridos comprobando cada uno de los bloques de encabezamiento de la cinta. Cuando se formatea un wafer, el sistema operativo divide la cinta en bloques, cada uno con su propia sección de nombre de archivo. Cuando se requiere cargar un archivo en el ordenador, busca hasta hallar el bloque que contenga la primera parte del archivo, lo carga y después busca el segundo.

Del mismo modo, cuando se solicita visualizar un directorio del wafer, el QOS lee cada uno de los nombres de archivo a medida que el cabezal de lectura/escritura va pasando por la cinta, y registra los bloques que contienen archivos y aquellos que están vacíos. Cuando se han leído todos los nombres de archivo, el sistema visualiza la lista de archivos más la cantidad total de espacio disponible expresada en bytes.

Gestión de archivos

La FMU, utilidad para gestión de archivos, es un conjunto de rutinas activadas por menú que cubren aplicaciones tales como formateo y lectura del directorio del wafer. Asimismo, contiene rutinas para copiar que permiten transferir datos desde disco, cassette o wafer a un wafer de seguridad. Ésta es, obviamente, una importante característica del sistema Quick Data Drive, porque pocos programadores pensarían en adquirir un sistema de almacenamiento, independientemente de lo bueno que fuera, que no pudiera transferir programas existentes al nuevo soporte.

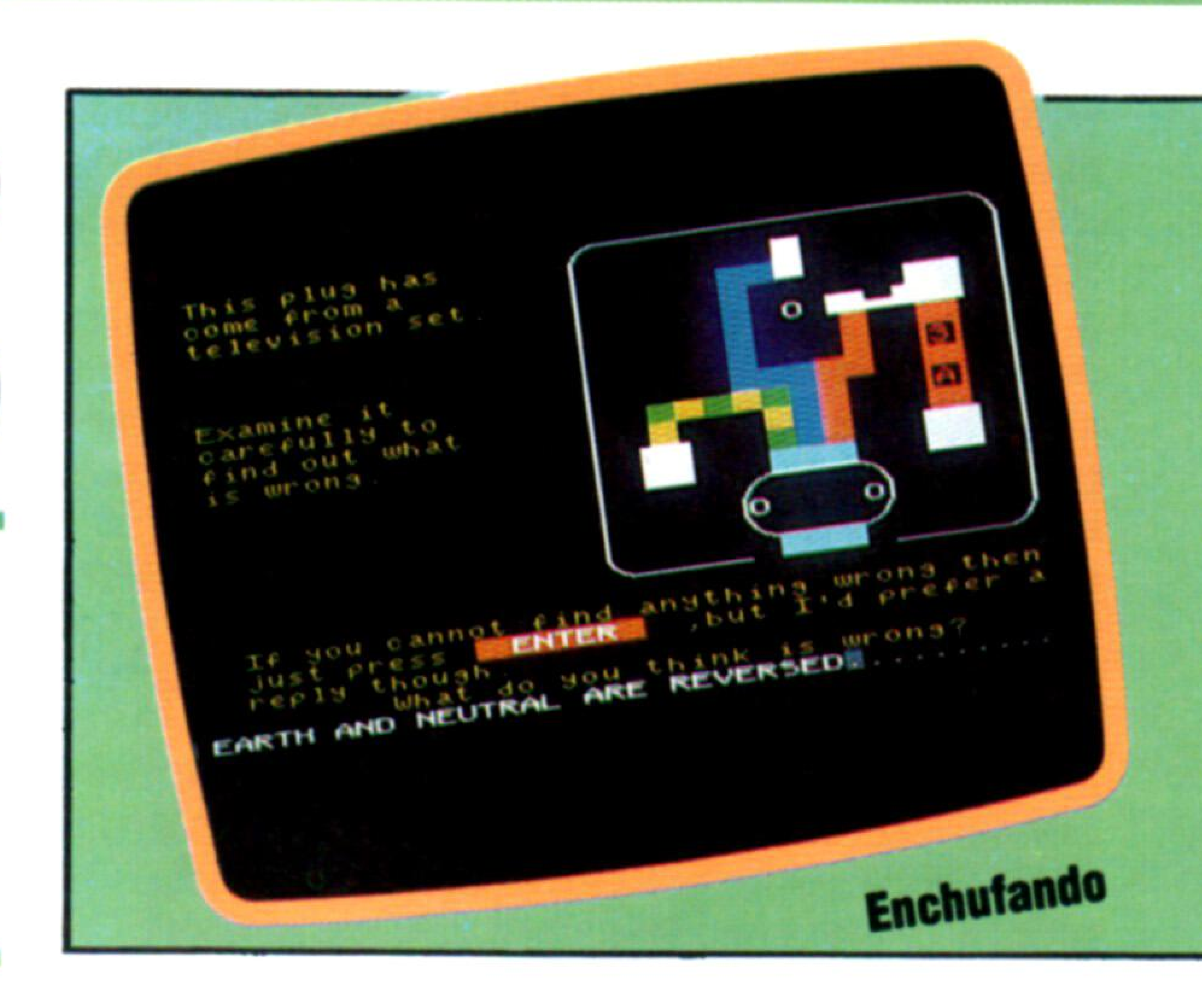
Por supuesto, el sistema tiene sus inconvenientes. Si bien las rutinas para copiar funcionan bien para programas en BASIC y archivos secuenciales, ciertas rutinas (en especial en código máquina) que se cargan en una zona específica de la memoria suelen causar problemas. Ello se debe a que las dos zonas que utilizan el QOS y la FMU son también las zonas en que se conserva el código máquina. Por lo tanto, cuando se cargan programas en código máquina, éstos se graban sobre el sistema operativo de la Quick Data Drive y el programa queda colgado.

Esta característica de diseño significa que si bien se pueden copiar los programas del usuario, aún no es posible la transferencia de software comercial. No obstante, de venderse suficientes cantidades de dispositivos Quick Data Drive, sin duda se producirá un programa para permitir dicha transferencia.

Hasta entonces queda en manos de Dean Electronics el persuadir a las casas de software de que vendan sus juegos y aplicaciones en wafers. En este sentido, quizá la Rotronics Wafadrive resulte ser un valioso aliado: las empresas de software podrán adquirir, entonces, cantidades mayores de wafers con mayores descuentos, cuando descubran que pueden colocar tanto su software para el Commodore 64 como para el Spectrum en el mismo soporte de almacenamiento.

Código que crea código

"Last One" y "Sycero" son dos generadores de programas que ejemplifican la situación actual del software que escribe software



Enchufando

Todos los programadores recién iniciados (y probablemente también los más experimentados) sueñan con la existencia de un programa que elimine de la programación el trabajo pesado y la frustración y acabe con unas líneas de código que se ejecuten perfectamente, a la primera, sin ninguna clase de depuración ni mensajes de error.

En cierto sentido, por supuesto, quien escribe un programa en BASIC ya está utilizando un programa para escribir programas: bien resida en ROM o bien haya que cargarlo desde disco o cinta, el intérprete de BASIC toma las líneas que digita el usuario y las reescribe en código máquina para que el ordenador pueda comprenderlas. Éste es un proceso mucho más sencillo que entrar los propios dígitos del código máquina, desprovistos aparentemente de cualquier significado.

La mayoría de los programadores estarán familiarizados con el truco de hacer que un programa genere líneas de programa extras ya sea utilizando el buffer del teclado con un contador creciente para números de línea, o bien colocándolas (POKE) en la memoria. Esto con frecuencia lo emplean los programadores de código máquina para atisbar (PEEK) las posiciones de memoria y escribirlas en un programa cargador en forma de sentencias DATA.

Algunos lenguajes, por ejemplo el COMAL, poseen comprobación de errores incorporada y se niegan a aceptar una línea de código que no se haya entrado de la forma correcta, de modo que usted no podrá colar un PRNT cuando lo que quiere significar en realidad sea PRINT. El FORTH rechazará una instrucción que emplee una palabra que no esté incluida en su vocabulario básico o que el usuario no haya definido previamente. El LOGO posee una capacidad similar para funcionar de acuerdo a sus propias limitaciones.

Sin embargo, las facilidades que acabamos de mencionar continúan sin permitir que el usuario, por ejemplo, le pida al ordenador que escriba un programa para calcular su impuesto sobre la renta. Pero el crecimiento de sistemas que imitan o parecen incorporar cierto nivel de inteligencia artificial aparece como una posibilidad menos remota. Por ejemplo, hay un programa llamado *Microtext*, que se puede utilizar para guiar a un usuario no técnico a través de complicadas tareas técnicas. También se lo puede emplear para generar ayudas interactivas para la enseñanza, programas que busquen fallos, cuestionarios y software para recuperación de información. Mediante el empleo de un sistema de "autor" para crear una aplicación, disponiendo durante la comprobación de un editor en pantalla activo, el software resultante se puede "publicar" y convertir en un sistema de sólo ejecución que el usuario no pueda modificar.

Por lo general el *Microtext* sólo está a disposición de las máquinas CP/M más costosas (haciendo de la disponibilidad de un sofisticado editor de pantalla una valiosa facilidad), pero también ha salido al mercado a un precio muy inferior para el Tatung Einstein.

"The Quill" e "Illustrator"

Otro generador de programas notablemente refinado es *The Quill* (La pluma), diseñado específicamente para quienes escriben juegos de aventuras. Permite que usted prepare y edite una base de datos de información (escenarios, objetos, cómo se los puede manipular, etc.) y la incorpora a un programa en lenguaje assembly que se ejecuta de buenas a primeras sin quedar colgado.

A *The Quill* recientemente se le ha unido *Illustrator*, que añade gráficos a las aventuras de texto. *The Quill* está destinado al Spectrum, el Amstrad 464, el Commodore 64 y el Oric, e *Illustrator* al Spectrum y al Amstrad, y pronto se editará también para Commodore. A pesar de su bajo costo, *The Quill* es lo suficientemente bueno como para haber sido utilizado para escribir muchos juegos de aventuras comerciales, entre los que destaca en particular *Hampstead*, de Melbourne House.

Aunque técnicamente ambos son generadores de programas, para la industria del micro existen sólo dos programas que realmente escriben programas: *The Last One* (El último), de nombre más bien ostentoso y que en el momento de su lanzamiento se anunció como el último software que usted necesitaría comprar en toda su vida, y el *Sycero*, bastante más potente aunque algo menos amable.

The Last One, o *TLO*, como prefieren llamarlo sus distribuidores, se escribió originalmente para máquinas CP/M de 64 K. Esto es evidente, ya que posee la misma clase de alusión constante a los *overlays* de disco que hacen que el *WordStar* sea tan lento. Ahora existe para máquinas MS-DOS con un mínimo de 256 K de RAM, pero si bien ello permite la adición de muchas facilidades más sofisticadas, continúa operando a un ritmo ligeramente lento. Sus autores afirman haber incorporado en el programa un cierto grado de inteligencia artificial.



Juegos de generación
Estas pantallas ilustran las clases de programas que se pueden producir con el *Microtext*. El programa de comprobación de enchufes, a la izquierda, formula al usuario preguntas sobre sistemas eléctricos. El programa de la derecha pone a prueba su conocimiento del procedimiento correcto a seguir cuando uno se encuentra atrapado en un hotel en llamas. Observe que ambos programas dependen de la interacción del ordenador con el usuario, comparando las respuestas del usuario con una base de datos de conocimientos retenida en su memoria. Esta clase de aplicaciones es ideal para los generadores de programas

The Last One: Para el Apple II y IIe, máquinas CP/M-80 y MS-DOS
Distribuido por: D J "AI" Systems Ltd, Summers Orchard, Speke Close, Ilminster, Somerset TA19 9BJ, Gran Bretaña
Microtext: Para el Tatung Einstein y máquinas CP/M-80
Distribuido por: Transdata Ltd, 11 South Street, Havant, Hants, Gran Bretaña
The Quill: Para Spectrum, Commodore 64, Amstrad 464 y Oric
Distribuido por: Gilsoft, 30 Hawthorne Road, Barry, South Glamorgan CF6 8LE, Gran Bretaña
Sycero: Para máquinas MS-DOS
Distribuido por: System C Ltd, 7 Mill Street, Maidstone, Kent ME15 6XW, Gran Bretaña

Ciertamente, si usted lo carga con un tipo de BASIC distinto del que está buscando, identificará correctamente el disco DOS en cualquier mensaje de error.

Aunque similares a nivel superficial (en tanto y en cuanto ambos son generadores de programas), *Sycero* y *The Last One* en realidad son bastante diferentes en cuanto a concepción, esfera de acción y la forma en que abordan el problema de esclarecer las necesidades del usuario con el fin de producir código utilizable (y transportable). Asimismo, son similares en el hecho de que ambos generan programas en BASIC y que son particularmente idóneos para la entrada, proceso, almacenamiento, recuperación e impresión de datos. Los dos se diferencian de otros programas que realizan sofisticadas funciones de base de datos, porque el código resultante se ejecuta independientemente del sistema generador.

En cierto sentido, *TLO* está dirigido a las necesidades de los usuarios de hoy en día, porque da por sentado (probablemente de forma correcta) que usted prefiere hacer todo el trabajo en pantalla, prescindiendo de preparativos y cálculos en hojas de papel. Pero utilizando *Sycero* y *TLO* de esta manera no se aprovecha cabalmente su potencial. El plan de acción de *Sycero*, de siete puntos, hace hincapié en la importancia de la planificación preliminar:

- Planificar
- Especificar sistema
- Dibujar pantallas
- Comprobar datos
- Definir programa
- Producir salida impresa
- Generar programa

Y tanto *Sycero* como *TLO* resaltan el punto con suficiente claridad en sus manuales: la organización y la planificación preliminares son esenciales para una buena programación. El *Sycero* afirma:

Corregir errores originados por un mal diseño puede llevar más tiempo que escribir el sistema en primer lugar. Siempre es tentador dedicar cinco minutos a pensar en un problema y luego, con los discos Sycero en la mano, correr hacia el ordenador y elaborar el sistema sobre la marcha. Éste no es un buen enfoque.

...usted siempre debe comenzar elaborando una lista exhaustiva de todo lo que desea que el ordenador produzca para usted, desde el punto de vista de lo que quiere ver en la pantalla (información on-line*) y de lo que desea en cuanto a salidas impresas ("listados"). Sólo tras haber determinado lo que usted espera del sistema, habrá de decidir qué información necesita para producir esos resultados.*

TLO es menos específico pero insiste en el mismo punto:

Planifique el flujo global de su programa y anticipe los errores que va a cometer el usuario final. Después prepárelo todo para hacer frente a los mismos de forma segura. De hecho, diseñe su programa tan concienzudamente como planificaría cualquier otra tarea. Por regla general, es más fácil comenzar escribiendo algunos programas cortos, unidos entre sí mediante un menú sencillo que llame a cada uno de ellos por turno cuando así sea necesario; más adelante, hallará que este enfoque modular permite planificar y crear con gran facilidad programas extensos y complicados.

Ninguno de estos paquetes generará sólo una clase de programas. No producirán ningún tipo de juego, por supuesto (ni siquiera una aventura basada sólo en texto), ni tampoco generarán un procesador de textos ni una hoja electrónica. Sus puntos fuertes residen en permitir que usted manipule virtualmente cualquier clase de datos, efectúe cálculos con ellos (que puede ser cualquier cosa, desde una simple extracción del IVA de una suma bruta hasta una larga y compleja ecuación de ingeniería), almacenarlos, recuperarlos e imprimirlos. En este sentido, ambos son sumamente poderosos.

Es materia de controversia el hecho de que trabajen o no tan bien como lo hace un programa directo de administración de bases de datos como el *dBase II*; pero es innegable que son más fáciles de utilizar y más amables que un programa de esta clase, cuya hostilidad suele ser directamente proporcional a su potencia.

Procesar listas

Basado en listas que pueden representar tanto datos como funciones, el LISP permite al programador adaptar el lenguaje a casi cualquier aplicación. Iniciamos una serie dedicada al LISP, en la que nos referiremos al uso que hace de las listas y veremos por qué ha tenido tanta aceptación en el campo de la inteligencia artificial.

El LISP ha tenido bastante difusión en los últimos años, fundamentalmente a raíz de su inclusión en la investigación y desarrollo en el campo de la inteligencia artificial. A medida que se acrecentaba el interés por él, se hizo evidente que el LISP podía actuar como un lenguaje para fines generales con una amplia variedad de aplicaciones.

Esta convicción generó una ingente cantidad de implementaciones del LISP con muchas aplicaciones diversas, desde la escritura de sistemas operativos y compiladores hasta la escritura de juegos de aventuras. Una lamentable consecuencia de esta proliferación del lenguaje es que, como sucede con tantos otros lenguajes, existen ahora muchos dialectos de LISP. A pesar de ello, la mayoría de las implementaciones están moderadamente estandarizadas (debido a su estructura simple y directa) y por lo general traducir el LISP de una máquina a otra es bastante sencillo.

Es probable que al utilizar el LISP advierta la carencia de una cierta función o instrucción. Esto se debe principalmente a la falta de un estándar oficial, con lo que sólo quedan las funciones que los programadores consideran más importantes. Sin embargo, tal como veremos, ampliar el lenguaje es extraordinariamente fácil, mediante la adición de las instrucciones nuevas que usted requiera.

El LISP es completamente diferente de los lenguajes más comunes, como el PASCAL, el FORTRAN y el BASIC. Posee una estructura sintáctica única, muy simple y uniforme. Esta estructura lo hace ideal para la manipulación de datos y problemas de emparejamiento.

La estructura inherente de las listas se puede adaptar fácilmente a las principales estructuras de información para ordenador, permitiendo utilizar el lenguaje para búsqueda y clasificación, funciones aritméticas e incluso para practicar juegos. Además, la mayoría de los microordenadores soportan instrucciones especiales para hacer uso de sus propias facilidades de sonido y de gráficos. Éste es, ciertamente, el caso del LISP Acornsoft para el BBC Micro, que es el que utilizaremos a lo largo de esta serie. No se preocupe si posee una versión diferente del lenguaje, ya que encontrará que todo se puede transferir directamente.

La base del LISP, como cabría esperar, es la estructura de datos de *lista*: el nombre del lenguaje proviene de los términos *"list processing"* (proceso de listas). En muchos sentidos, una lista es similar a la más familiar "matriz" de la mayoría de los micros. No obstante, a diferencia de la matriz, una lista no posee ninguna longitud específica. Cualquier lista de elementos se indica encerrándola entre paréntesis, en la forma:

```
(a b c d e ...)
```

donde a, b, c, etc., son los elementos de la lista. El término técnico para estos elementos es *átomos*, y pueden ser datos numéricos, no numéricos (datos de caracteres o variables) o incluso otra lista. Observe que los elementos separados de una lista no se separan entre sí mediante comas; en cambio, se utilizan espacios.

Las operaciones se realizan sobre las listas utilizando funciones, de forma muy similar a una función de BASIC (instrucción DEF FN), que lleva a cabo una operación sobre sus argumentos para producir un resultado. Una función se escribe de la forma:

```
(func a b c d ...)
```

donde func es el nombre de la función y a, b, c, etc. son sus argumentos. Como puede ver, la función tiene un aspecto muy parecido al del primer elemento de la lista. De modo que, por ejemplo, una lista compuesta por los seis primeros números primos se escribiría:

```
(1 2 3 5 7 11)
```

Si quisiéramos sumar estos números en BASIC, utilizaríamos:

```
1+2+3+5+7+11
```

En LISP, aplicaríamos la función plus y escribiríamos:

```
(plus 1 2 3 5 7 11)
```

que nos devolvería como respuesta 29. Una de las características de la función PLUS es que puede tomar cualquier cantidad de argumentos; de modo que podríamos igualmente escribir:

```
(PLUS 1 2 3 (PLUS 5 7 11))
```

Aquí, primero se evalúa la función PLUS más interior, que devuelve la respuesta 23. Ésta se utiliza luego como el cuarto argumento del PLUS más exterior para devolver la respuesta final 29. Un punto importante a observar aquí es la facilidad con que se pueden anidar las funciones. Anidando otra función, SETQ, podemos asignar el resultado a la variable A:

```
(SETQ A(PLUS 1 2 3 5 7 11))
```

Implementaciones del LISP
A pesar del creciente interés que existe por el LISP, en la actualidad hay pocas versiones del lenguaje al alcance del presupuesto de los usuarios de un micro. Existe el LISP Acornsoft para el BBC Micro y el Acorn Electron a un precio razonable, si bien la mayoría de los usuarios se encontrarán con que tendrán que adquirir la Guía para el Usuario del LISP, que suma otro coste adicional.

Otros micros no están tan bien provistos. Existen, no obstante, algunas versiones CP/M del lenguaje, en particular Toolworks LISP/80 e iLISP. Hay otra versión CP/M, muLISP-83, de Microsoft, que es considerablemente más cara.

Los intérpretes basados en CP/M mencionados anteriormente se ejecutarán en la mayoría de los micros personales que soporten CP/M (Memotech, Einstein, etc.), aunque los usuarios del Amstrad no podrán utilizarlos debido a la falta de espacio de memoria (el mínimo requerido es de 48 K). Probablemente la mejor relación calidad-precio la ofrezca Toolworks, con un espacio de almacenamiento de aproximadamente 3 600 celdas de listas y 11 000 caracteres para nombres de átomos en 48 K. Todos los programas se pueden adquirir en Grey Matter Ltd, 4 Prigg Meadow, Ashburton, Devon, TQ13 7DF, Gran Bretaña



donde SETQ posee dos argumentos: la variable A y una función que halla el entero 29. Por supuesto, SETQ es en sí misma una función y por tanto debe devolver un resultado (en este caso el valor 29). En consecuencia, la siguiente asignación, que es ilegal en la mayoría de las versiones de BASIC:

```
LET B=1+2+3+5+7+11
```

se podría escribir en LISP como:

```
(SETQ B(SETQ A(PLUS 1 2 3 5 7 11)))
```

Introduciendo la función TIMES, podemos asignar, por ejemplo, 2 veces el valor de A a la variable B:

```
(SETQ B(TIMES 2(SETQ A(PLUS 1 2 3 5 7 11))))
```

En primer lugar se calcularía el PLUS para devolver el valor 29, que se le asignaría a la variable A en virtud de la función SETQ más interior. Esta función, a su vez, daría como resultado 29, a utilizar como el segundo argumento de la función TIMES. El nuevo resultado, 58, se pasaría entonces a la función SETQ más exterior para colocar el valor en la variable B. Toda la expresión daría como resultado 58, que se emplearía, entonces, para otras funciones, y así sucesivamente.

Observe que los paréntesis están proliferando mucho en el último ejemplo. Ésta es una característica del LISP, y seguir el rastro de todos estos paréntesis puede ser bastante tedioso, aunque, si el programa está bien diseñado, tienden a ocuparse de sí mismos. Además, algunos sistemas de LISP ofrecen la ayuda de informar sobre la cantidad de paréntesis que posee.

Por ejemplo, cuando se utiliza el LISP Acornsoft en el BBC Micro, la cantidad de flechas que aparecen al comienzo de la indicación de línea señala la cantidad de paréntesis que es necesario cerrar para completar la expresión.

En la última expresión introdujimos la función TIMES y le dimos dos argumentos: el valor entero 2 y una expresión de lista que previamente había dado como respuesta 29. Sin embargo, TIMES, al igual que PLUS, puede tener un número variable de argumentos. Por tanto, todas las expresiones siguientes serían legales:

```
(TIMES 1 2 4 8 16)
(TIMES 1 2 4 8 (TIMES 4 4))
(TIMES 1 2 (TIMES 2 2)(TIMES 2 4)(PLUS 8 8))
```

y todas devolverían el resultado 1024.

En realidad, la mayoría de las implementaciones del LISP poseen un límite en cuanto a la cantidad de argumentos que pueden tener este tipo de funciones. Por ejemplo, el LISP Acornsoft tiene un límite de 28 argumentos, y algunas otras versiones tienen restricciones aún más severas.

Llegados a este punto, podemos ver que una instrucción en LISP es simplemente una lista de elementos, en la cual el primero es la función a realizar, y los subsiguientes son los argumentos de la función. Éstos, a su vez, pueden ser listas con el primer elemento de cada una siendo una función que devuelva un resultado. Ahora se plantea la cuestión de qué hacer si no deseamos ninguna función. Es posible que querramos meramente preparar una lista de elementos de datos, supongamos, para utilizarla como un título.

MATRIZ (DIMensionada en n)

1	2	3	4	5	6	7	8	//			n

LISTA (sin longitud determinada)

1 → 2 → 3 → 4 → 5 → etc.

Listas contra matrices
Las listas ofrecen dos ventajas principales sobre las matrices. En primer lugar, no hace falta reservar memoria para ellas antes de que se la necesite; en segundo lugar, una lista es una estructura "dinámica", lo que equivale a decir que no necesita tener longitud fija y que se puede comprimir o expandir para acomodar datos durante la ejecución del programa. Además, las listas se prestan más fácilmente a los procedimientos recursivos que se utilizan con frecuencia en la programación de inteligencia artificial

No podemos escribir:

```
(MI COMPUTER EL MEJOR)
```

como una lista de cuatro elementos de datos, porque el LISP intentaría evaluar MI como el nombre de una función, con tres argumentos, y pensaría que cada uno de los tres argumentos (COMPUTER, EL y MEJOR) son tres nombres de variables.

Podemos decir al LISP que no evalúe una expresión anteponiéndole un apóstrofo ('), de modo que escribiríamos lo anterior como:

```
'(MI COMPUTER EL MEJOR)
```

Entonces podríamos asignarle esta lista a una variable:

```
(SETQ REV '(MI COMPUTER EL MEJOR))
```

que le asignaría a la variable REV una lista de cuatro elementos no numéricos.

Observe que en LISP no hay ningún tipo de variable.

Si estuviéramos utilizando BASIC, una variable de enteros asumiría la forma A%, una variable de reales A y una variable en serie A$. En LISP las variables no se diferencian de esta manera, de modo que todas las expresiones siguientes son legales:

```
(SETQ A 3)                A=el entero 3.
(SETQ A 'COMPUTER)        A=la serie 'COMPUTER'.
(SETQ A (PLUS 2 4 8))     A=el resultado de 2+4+8.
(SETQ A '(1 2 4 8))       A=la lista (1 2 4 8).
(SETQ A '(MI COMPUTER))   A=la lista (MI COMPUTER).
(SETQ A B)                A=el valor de la variable B.
```

Si bien los tipos de enteros y en serie son relativamente estándares, pocos microordenadores soportan una versión de LISP que efectúe aritmética de punto flotante. En la mayoría de los casos, la aritmética de enteros es suficiente y en el improbable caso de que fuera necesaria la aritmética de punto flotante, sin ninguna duda se podría simular empleando los enteros estándares.

En este punto puede parecer difícil comprender en qué medida puede ser de utilidad el lenguaje. En el próximo capítulo veremos las funciones del LISP que pueden manipular sus argumentos de datos, y veremos de qué forma se emplean los principios de la recursión.

Queda detenido

El Z80 ofrece tres modos de interrupción: veamos cómo el sistema operativo del Sinclair Spectrum vectoriza las interrupciones

Las facilidades basadas en interrupciones, propias del sistema operativo del Spectrum, son menos directas que las utilizadas en el BBC Micro. El empleo de las interrupciones en el Spectrum exige sumo cuidado. En caso de inhabilitarlas, o "desactivarlas" inadvertidamente, la máquina puede quedar gravemente afectada: el teclado puede no ser leído, por ejemplo, o puede colgarse el sistema durante la ejecución de programas en BASIC.

La CPU del Z80 es el corazón del Spectrum y responde a dos tipos diferentes de interrupciones: las enmascarables y las no enmascarables (NMI). Su diferencia es clara. Se puede programar la CPU de modo que ignore una señal de interrupción enmascarable, pero el procesador siempre responderá a una NMI (no enmascarable).

Estas interrupciones NMI son poco prácticas en el Spectrum, ya que la rutina ROM que gestiona las NMI muestra bastantes errores. La intención original de los diseñadores del OS era la de permitir al usuario especificar una dirección en las variables de sistema no utilizadas en las posiciones &55cb0 y &5CB1, donde se saltaría cada vez que la CPU recibiera una señal de NMI. Pero la respuesta habitual del Z80 es la ejecución del código máquina que comienza en la dirección &66, lo que se traduce en una inicialización completa del sistema. Por lo tanto, sólo nos ceñiremos al examen de las interrupciones enmascarables.

Modos de operación de interrupción

Varias son las formas en que el Z80 puede responder a una interrupción enmascarable, llamadas precisamente *modos* de operación de interrupción. Trataremos tan sólo aquellos modos importantes para el sistema operativo del Spectrum.

En el momento de la inicialización (que se da al conectar la máquina o al generar la instrucción NEW) la CPU establece el modo de interrupción 1, también conocido por IM1. Éste es el modo habitual en que funciona el Spectrum. Los impulsos de interrupción se llevan a la CPU por medio de la ULA Sinclair (*uncommitted logic array*: tabla lógica sin cometido) a una densidad de 50 interrupciones por segundo. En el modo IM1, la CPU ejecuta una instrucción del RST en &0038 al recibir la señal. Esto provoca un salto a las rutinas que leen el teclado y actualizan el contador FRAMES, situado en la RAM en las posiciones que van de la 23672 a la 23674 (estos tres bytes forman un contador de 24 bits que, por ende, se actualiza cada 20 milisegundos).

Una vez interpretada una línea del BASIC, el intérprete de este lenguaje espera una interrupción antes de pasar a interpretar la siguiente sentencia. Esto significa que si usted neutraliza las interrupciones, la ejecución de un programa en BASIC se parará.

La CPU ignorará una interrupción enmascarable una vez ejecutada la instrucción DI (D*isable* I*nterrupt*: desactivar interrupción), y volverá a tenerlas en cuenta después de la orden EI (E*nable* I*nterrupt*: reactivar interrupción).

Existen diversas rutinas en el OS del Spectrum que exigen la desactivación de interrupciones mientras son ejecutadas. Se trata, por lo general, de rutinas dependientes del tiempo tales como la rutina BEEP del generador de sonido y las rutinas para guardar (o cargar) datos en cinta. Las interrupciones pueden ser desactivadas momentáneamente por medio de la impresora ZX, la Interface 1 o el microdrive. Es claro que el OS vuelve a activarlas una vez acabada la rutina.

Un resultado práctico de las diversas rutinas que desactivan interrupciones es que mientras estas rutinas son ejecutadas el contador FRAMES no se incrementa. Por tanto, se "perderá tiempo" mientras son ejecutadas estas operaciones. Dado que una interrupción puede darse mientras se está ejecutando

Anticiparse a todas las posibilidades

Aunque en muchos casos se puede asumir que el bus de datos del Spectrum contiene, cuando se genera una interrupción, el valor 255, algunos periféricos (como la interface de palanca de mando Kempston) pueden alterar este valor. La forma más fácil de obviar el problema es cerciorarse de que no se está empleando el modo 2 (IM2) con periféricos acoplados. Otro método, abierto a todo valor posible en el bus de datos, consiste en llenar una página de memoria con valores iguales. Se carga después el número de página en el registro I antes de seleccionar el IM2. Cuando se da una interrupción, el Z80 toma la dirección de la rutina de servicio de interrupciones sacándola de alguna posición de la página especificada: la posición exacta que será determinada por el valor que lleve el bus de datos. Primeramente cargamos el registro I con el valor &FC, y llenamos los bytes &FC00 al &FD00 con el valor &FB. Después seleccionamos el IM2. Si el bus de datos contiene &C3 cuando ocurre la interrupción siguiente, el Z80 sacará la dirección de la rutina de servicio de la posición &FCC3: la parte &FC de la dirección ha sido suministrada por el registro I. La dirección tomada será &FBFB, ya que todos los 256 bytes de la página FC (añadiendo el byte cero de la página &FD) han sido llenados con el mismo valor. Hay dos puntos a tener en cuenta. Primero: su rutina de servicio de interrupciones (ISR) tendrá que estar siempre en una dirección con idénticos bytes *hi* y *lo* (p. ej., &C4C4 o bien &FDFD). Segundo: debe recordar que el bus de datos puede que contenga el valor &FF en el momento de tener lugar la interrupción. En este caso el Z80 buscará la rutina ISR en las direcciones nnFF (byte *lo*) y (nn+)00 (byte *hi*), donde nn es el valor del registro I. Por esta razón, debe recordar poner el primer byte de la página siguiente



uno de sus programas, será oportuno que desactive las interrupciones para el fragmento de programa en que la temporización exacta sea importante. Pero, recuerde, es *esencial* que las reactive antes de volver al BASIC.

Interrupciones vectorizadas

Para hacer un uso práctico de las interrupciones del Spectrum, es necesario cambiar el modo y poner aquel que ofrezca mayor versatilidad. El modo de interrupción más útil es el IM2, cuyo funcionamiento es mucho más complicado que el de IM1. Mientras el IM1 siempre consiste en un salto a la dirección &0038 (empleando la instrucción RST &0038), el IM2 puede saltar a cualquier rutina de la memoria. La dirección a la cual salta la CPU se especifica por medio del llamado *vector de interrupción*.

En una interrupción vectorizada, el vector retiene la dirección de la *rutina de servicio de interrupciones* que se ejecutará cuando se dé una interrupción enmascarable. La CPU conoce dónde está situado el vector en la memoria gracias a un registro especial del Z80 llamado *registro I*. La dirección de la rutina del vector de interrupción se halla combinando el contenido del registro I con el del bus de datos en el instante de la interrupción. En algunos sistemas el dispositivo que origina la interrupción colocará un byte en el bus de datos para avisar a la CPU cuál fue la causa de la interrupción.

Pero la ULA del Sinclair no pone ningún valor en el bus de datos cuando envía una señal de interrupción al Z80, aunque la forma en que está configurado el hardware del ordenador permite que el bus de datos contenga el valor 255 cuando no se ha aplicado otra entrada en él. De esta manera, el vector de interrupción siempre estará situado en una frontera de página de memoria, con el byte *lo* de la rutina de servicio contenido en la dirección &nnff y el byte *hi* en la dirección &(nn+1)00, siendo nn el contenido del registro I.

Por ejemplo, si el registro contiene el valor &FB, la dirección del vector estará en &FBFF. El byte *lo* del vector, en la posición &FBFF, contendrá el byte *lo* de la dirección de la mencionada rutina de servicio, y el byte *hi* del vector, en la dirección &FC00 contendrá el byte *hi* de la dirección de la ISR.

Existen algunas reservas sobre el posicionamiento en memoria de la ISR: así, por ejemplo, los primeros 16 Kbytes están asignados a la ROM, y por ello no pueden usarse. Por otro lado, algunos problemas del hardware impiden la operación correcta del OS con un valor del registro I que se comprenda entre el 64 y el 127.

Empleo del modo 2

Examinemos ahora cómo se pone el Z80 en el modo 2, y se establece el vector de interrupción con la dirección de la ISR que ha de emplearse:

```
F3              di                    ;desactiva interrupciones
210000          ld    h1,ADDRESS      ;toma dirección ISR en HL
22FFFB          ld    (#FBFF),hl      ;pone dirección en vector
3EFB            ld    a,#FB           ;byte hi del vector...
ED47            ld    i,a             ;...al registro I
ED5E            IM    2               ;establece modo 2
FB              ei                    ;reactiva interrupciones
C9              ret                   ;vuelta al BASIC
```

Naturalmente este listado presupone que hay una rutina en ADDRESS que gestiona las interrupciones; de lo contrario, es posible que todo se venga abajo. Además, toda rutina de servicio de interrupción ejecutará idealmente todas las tareas llevadas a cabo por lo general en el Spectrum en su modo habitual de interrupción. Esto se hace mejor con una llamada a la rutina en la dirección &38. El siguiente programa en lenguaje assembly cambia el modo de interrupción al IM2 y obliga a la CPU a ejecutar la rutina de ADDRESS a cada interrupción (en este caso sólo ejecuta las funciones habituales que realiza el Spectrum en su modo normal).

```
                org   60000           ;especifica dir inicio
        vector: equ   #FEFF           ;FEFF es la dir vector
216FEA  change: ld    hl,addres       ;lleva address a HL
22FFFE          ld    (vector), hl    ;establece vector
F3              di                    ;desactiva interrupciones
3EFE            ld    a,#FE           ;establece el...
ED47            ld    i,a             ;...registro I
ED5E            im    2               ;cambia modo interrups.
FB              ei                    ;reactiva interrupciones
C9              ret                   ;vuelta al BASIC
F3      addres: di                    ;desconecta interrupciones
FF              rst   #38             ;procedimiento IM1 normal
FB              ei                    ;reactiva interrupciones
C9              ret
```

Si se llama a la rutina en la dirección CHANGE, se establecerá el nuevo modo de interrupción y el vector. Una vez conseguido esto, la rutina en ADDRESS hará su trabajo cada cincuentavo de segundo. Por el momento esta rutina no hace nada especialmente útil, pero pronto le asignaremos alguna función de interés. Si se necesita cambiar el modo de interrupción al normal en cualquier momento del programa, podemos emplear una rutina como la que sigue para que realice esto:

```
F3              di
3E3F            ld    a,#3f           ;restablece reg I
ED47            ld    i,a             ;...a su valor normal
ED56            im    i               ;modo normal
FB              ei                    ;reactiva interrupciones
C9              ret
```

Para hacer que una rutina de servicio de interrupción ejecute algún código máquina escrito por nosotros, trataremos ese código como una subrutina y lo llamaremos (CALL) de esta manera:

```
F3      addres: di
F5              push af               ;salva registros...
C5              push bc
D5              push de
E5              push hl
FF              rst #38               ;llama ISR normal
F3              di                    ;ISR hizo EI, por tanto DI otra
                                      vez
CD0000          call PROG_ADD         ;llama nuestra rutina
E1              pop hl                ;restaura registros...
D1              pop de
C1              pop bc
F1              pop af
FB              ei
C9              ret
```

Naturalmente, si la rutina de servicio es más bien larga, ralentizará ligeramente al Spectrum, lo que afectará a la ejecución del programa y a la velocidad en que se incrementa FRAMES. Por último, recuerde siempre que debe desactivar las interrupciones durante su rutina para evitar que ocurra una segunda interrupción a mitad del tratamiento de la primera.

Programa "Música bajo interrupciones"

El siguiente programa muestra cómo pueden usarse las interrupciones para obtener "música de fondo" en el Spectrum. El Compilador de Datos de Notas es un programa en BASIC por el que se pueden escribir melodías diversas, mientras que el listado assembly (o el Cargador en BASIC si no se dispone de un ensamblador) le permitirá establecer la rutina musical basada en interrupciones y pasarle los datos compilados por el Compilador de Datos de Notas. Si se ejecuta el Compilador de Datos de Notas se nos presenta un menú con tres opciones. Si optamos por 'C-COMPILE' se convertirá la melodía (contenida en las sentencias data que están entre las líneas 10 y 900) en un fragmento de código máquina utilizable por la rutina de interrupción. La opción 'P' hará sonar la melodía de modo que pueda realizar los cambios que le apetezcan. Mientras 'R' volverá al BASIC para que pueda guardar (SAVE) los datos compilados o cambiar los datos musicales contenidos en las sentencias data. Obsérvese que si se emplea la opción 'C', se obtendrá una interrogación para que especifiquemos dónde se desean almacenar los datos de notas en la memoria. Cerciórese de que los almacena por encima de RAMTOP, cuyo valor habrá alterado previamente para dar espacio a los data. Una vez compiladas las notas y almacenadas, la dirección de base y la longitud del código quedarán visualizados. Para guardar el código, vuelva al BASIC y teclee: SAVE "NOTEDATA" CODE (dirección de base), (longitud en n.º de bytes). Por último, si desea emplear sus propias melodías, elimine las líneas que van de la 10 a la 90 y escriba sus propios datos entre estas líneas. Digite el listado assembly por medio de un ensamblador y guárdelo en cinta. Si no posee un ensamblador emplee el Cargador del BASIC. La rutina tiene una dirección de base en 65021, por lo tanto cerciórese de que antes de cargar el código en la memoria usted ha rebajado convenientemente el RAMTOP (bastará con CLEAR 65000). Una vez ensamblado y cargado el código, será necesario cargar el código compilado por el Compilador de Datos de Notas (ver más abajo). De nuevo no olvide de rebajar el RAMTOP para dar espacio a los datos de las notas. El programa *Música bajo interrupciones* exige el conocimiento exacto de la posición de los datos de notas que ha de hacer sonar, para que funcione. Esta información se puede pasar a la rutina por medio del siguiente programa en BASIC:

```
10 LET L=65152 : LET
   V=(*direccion base de
   los datos de notas*)
20 POKE L+1,INT(V/256):
   POKE L,V-PEEK
   (L+1)*256
```

Hecho esto, RAND USR 65041 establecerá el IM2 y sonará la melodía. RAND USR 65071 seleccionará el IM1 y cesará la música.

Compilador de Datos de Notas

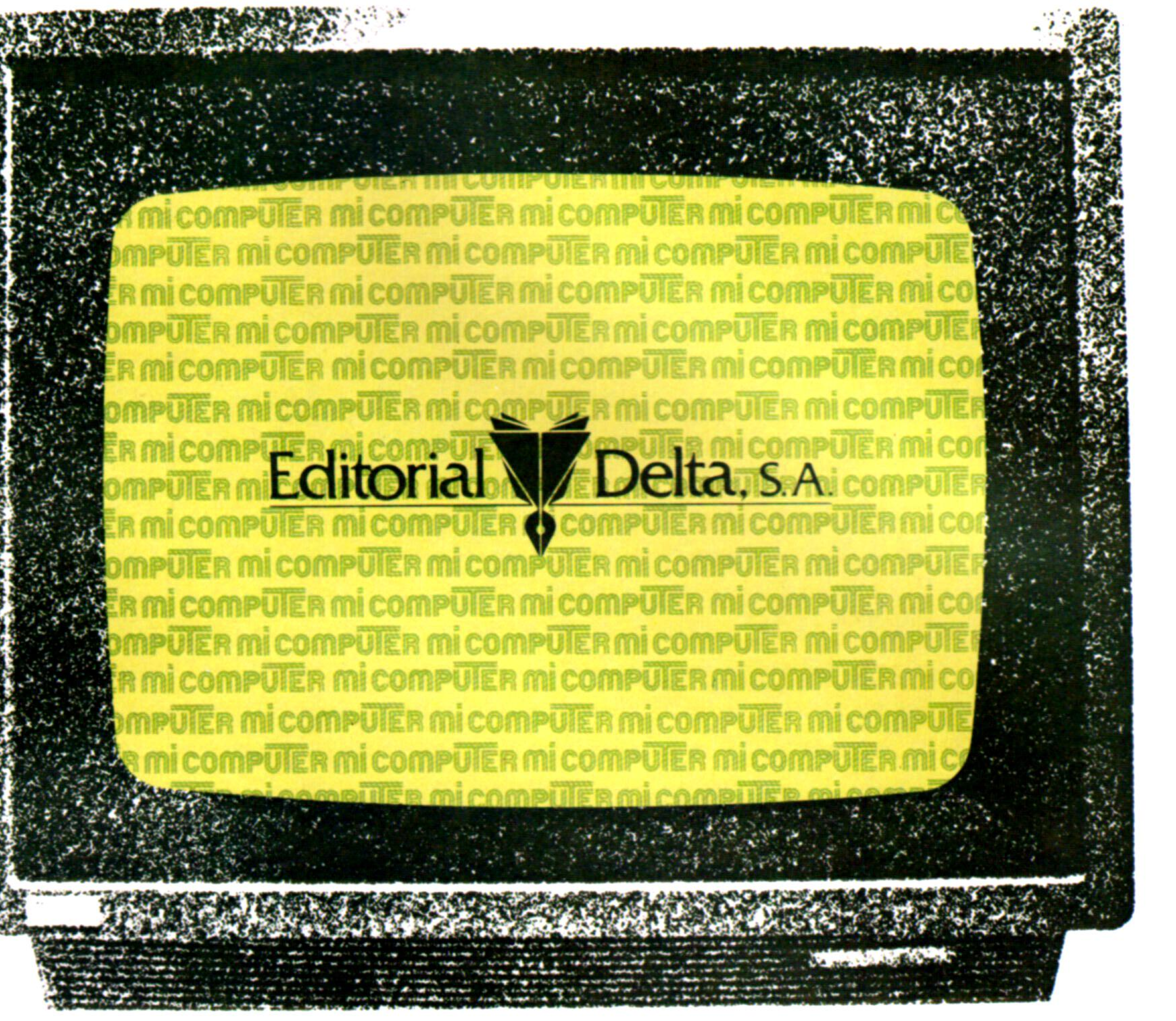
```
   1 >>>MUSICA BAJO INTERRUPCIONES<<<
   2 >>Compilador Datos Notas<<
   3 REM
   6 REM *DATOS DE NOTAS EN FORMA BEEP DEL SPECTRUM
   7 REM
   9 RESTORE
  10 DATA 1,12,2,9,1,9,1,9,1,8,1,9,2,17,1,12,2,12,1,9,2,10,1,
     10,2,10,1,12,5,14
  20 DATA 1,14,2,7,1,7,1,7,1,6,1,7,2,16,1,14,2,14,1,10,2,9,1,
     9,2,9,1,10,5,12
  30 DATA 1,12,2,9,1,9,1,9,1,8,1,9,2,17,1,12,2,12,1,12,2,11,
     1,19,2,19,1,19,5,19
  40 DATA 1,17,2,16,1,19,1,19,1,18,1,19,2,14,1,19,1,19,1,18,
     1,19,2,12,1,11,2,12,1,11,3,12
  50 DATA 3,10,1,9,1,8,1,9,2,14,1,12,3.2,9,3.2,5,3.2,2,
     2,3.2,7,5,5
  60 DATA 1,5,1,7,1,9,1,10,2,16,1,14,3.2,12,3.2,17,3.2,16,
     3.2,14,5,12
  70 DATA 1,12,2,14,1,14,1,14,1,13,1,14,3,16,3,9
  80 DATA 2,17,1,17,1,19,1,17,1,19,5,21
  90 DATA 1,21,2,19,1,17,2,14,1,10,3.2,9,3.2,5,3.2,7,3.2,4,5,5
 997 REM
 998 REM NO BORRAR LINEA 999
 999 DATA 255,255
1000 REM
2000 REM **** MENU ****
2010 BORDER 1: PAPER 1: INK 6
2020 CLS
2030 PRINT AT 5,4;"COMPILADOR MUSICA";AT 10,5;"C —
     Compile note data";AT 12,5;"P — Play tune in beeps";AT
     14,5;"R — Return to BASIC"
2040 LET a$=INKEY$
2050 IF a$="c" OR a$="C" THEN GO TO 5000
2060 IF a$="p" OR a$="P" THEN GO TO 3000
2070 IF a$="r" OR a$="R" THEN CLS : STOP
2080 GO TO 2040
3000 RESTORE
3010 READ d,f: IF d=255 AND f=255 THEN GO TO 2000
3020 BEEP d/10,f-6: GO TO 3010
5000 RESTORE : CLS : INPUT "DIRECCIONE BASE DE DATA
     NOTAS? ";dd1: POKE 23301, INT (dd1/256): POKE
     23300,dd1-(256*PEEK 23301): CLEAR dd1-1: LET
     dd1=PEEK 23300+256*PEEK 23301
5005 LET f=0: LET d=dd1
5010 READ delay,pitch
5015 LET delay=INT (delay*6)
5020 LET freq=(1.0594631^pitch)*256.
5030 LET bip=INT ((437500/freq)-30.125)
5035 IF delay=255*6 THEN POKE d,255: GO TO 6000
5037 LET f=f+1: PRINT AT 10,10;"NOTA)";f
5040 POKE d,delay
5060 POKE d+1,bip-(INT (bip/256)*256)
5070 POKE d+2,INT (bip/256)
5080 LET d=d+3
5090 GO TO 5010
6000 PAUSE 50: CLS : LET len=(d+2)-dd1
6010 PRINT ''"NUMERO DE BYTES DE DATA DE NOTAS...";len
6020 INPUT "PULSE 'ENTER' PARA VOLVER A MENU  "; LINE a$:
     GO TO 2000
```

Listado assembly

```
                org   65021
0000    vect:   defw  0            ;2 bytes para el vector
DDE5    inter:  push  ix
E5              push  hl
C5              push  bc
D5              push  de
F5              push  af
C334FE          jp    start        ;salto a rutina musica
F1      enprg:  pop   af
D1              pop   de
C1              pop   bc
E1              pop   hl
DDE1            pop   ix
C33800          jp    56           ;salto a ISR de ROM
21FFFD  on:     ld    hl,inter     ;establece vector
22FDFD          ld    (vect),hl
F3              di
3EFD            ld    a,253
ED47            ld    i,a          ;toma byte hi de vect en I
ED5E            im    2            ;pone modo 2 interr
2A80FE  initi:  ld    hl,(datad)   ;apunta a data
227EFE          ld    (locat),hl
3E01            ld    a,1          ;switch=1=suena nota
3282FE          ld    (swtch),a
3D              dec   a            ;demora=0
3283FE          ld    (delay),a
FB              ei
C9              ret                ;vuelta al BASIC
F3      off:    di
ED56            im    1            ;pone modo 1 interr
FB              ei
                ret
C9      ;
3A82FE  start:  ld    a,(swtch)    ;toma switch en a
FE00            cp    0
CA08FE          jp    z,enprg      ;si switch=0,fin
3A83FE          ld    a,(delay)
FE00            cp    0
CA4BFE          jp    z,nwnot      ;si demora=0,sonar
3D              dec   a            ;demora=demora-1
3283FE          ld    (delay),a    ;almac. nuevo valor dem.
C308FE          jp    enprg        ;ir a rutina fin
2A7EFE  nwnot:  ld    hl,(locat)   ;toma posicion datos
7E              ld    a,(hl)       ;toma demora sig. en a
FEFF            cp    255          ;255=datos finalizados
CA6CFE          jp    z,reset      ;datos finalizados, restaurar
3283FE          ld    (delay),a    ;almacena nueva demora
23              inc   hl           ;apunta a sig. byte datos
7E              ld    a,(hl)       ;lo toma en a
23              inc   hl           ;apunta a sig. byte datos
5F              ld    e,a          ;almacena byte 1 en e
7E              ld    a,(hl)       ;almacena byte2 en a
23              inc   hi           ;apunta a byte sig.
227EFE          ld    (locat),hl   ;almacena dirección byte sig.
67              ld    h,a          ;toma datos frecuencia...
68              ld    l,e          ;...en HL
110400          ld    de,4         ;duración nota de 4
CDB503          call  949          ;llama rut. BEEP en ROM
F3              di                 ;esta rutina hizo EI, por tanto DI
C308FE          jp    enprg        ;ir a rutina fin
2A80FE  reset:  ld    hl,(datad)   ;restaura varios...
227EFE          ld    (locat),hl   ;...punteros
3E01            ld    a,1
3282FE          ld    (swtch),a
3D              dec   a
3283FE          ld    (delay),a
C308FE          jp    enprg        ;ir a rutina fin
0000    locat:  defw  0            ;reserva memoria...
0000    datad:  defw  0            ;...para varios punteros
00      swtch:  defb  0
00      delay:  defb  0
```

Cargador del BASIC

```
  10 RESTORE 9000
  20 CLEAR 64000
  30 LET cqs=0: FOR f=65021 TO 65156
  40 READ dat: POKE f,dat
  50 LET cqs=cqs+dat
  60 NEXT f
  70 IF cqs<>18850 THEN PRINT "ERROR EN DATA!!!" : STOP
  80 PRINT "OK CORRECTOS LOS DATA"
  90 STOP
9000 DATA 0,0,221,229,229,197,213,245,195,52,254,241,209,
     193,225,221,225,195,56,0,33,255,253,34,253,253,243,
     62,253,237,71,237,94,42,128,254,34,126,254,62,1,
     50,130,254,61,50,131,254,251,201,243,237,86,251,201,
     58,130,254,254,0,202,8,254,58,131,254,254,0,
     202,75,254,61,50,131,254,195,8,254,42,126,254,126,
     254,255,202,108,254,50,131,254,35,126,35,95,126,35,
     34,126,254,103,107,17,4,0,205,181,3,243,195,8,
     254,42,128,254,34,126,254,62,1,50,130,254,61,
     50,131,254,195,8,254,0,0,0,0,0,0,0,
```


mi computer
Editorial Delta, S.A.


10093

9 788485 822836